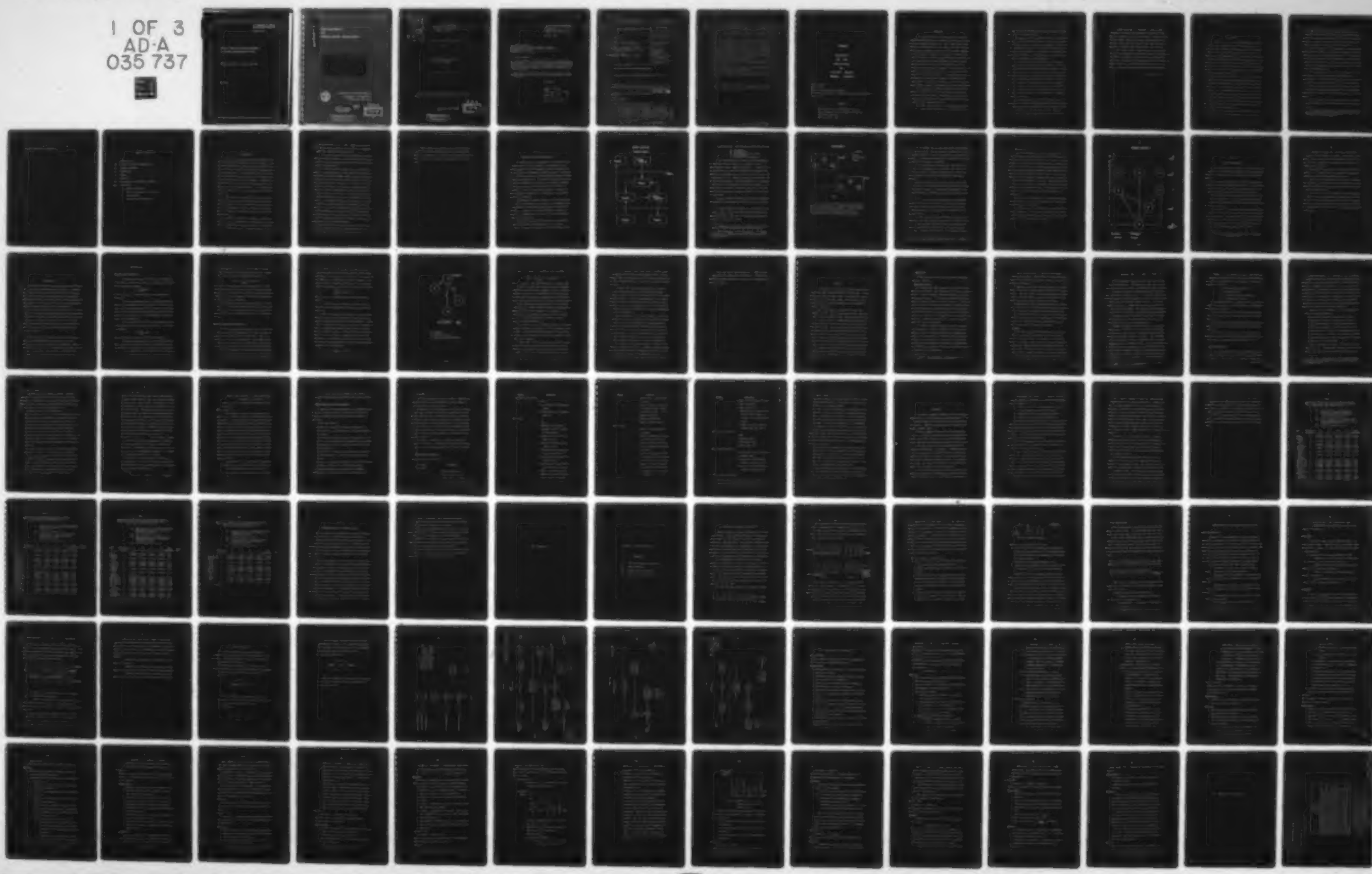
AD-A035 737 CORNELL UNIV ITHACA N Y DEPT OF OPERATIONS RESEARCH F/G 21/5
SPAERS: SIMULATION FOR THE PERFORMANCE AIRCRAFT ENGINE REPAIR S--ETC(U)
MAY 76 H GIVRAY, R SLON N00014-75-C-1172

UNCLASSIFIED NL

1 OF 3
AD-A
035 737



U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A035 737

SPAERS: SIMULATION FOR THE PERFORMANCE OF AIRCRAFT ENGINE REPAIR SYSTEMS

CORNELL UNIVERSITY, ITHACA, NEW YORK

MAY 1976

DEPARTMENT
OF
OPERATIONS RESEARCH

COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK

Master of Engineering Report

May 1976

SIMULATION FOR THE PERFORMANCE OF AIRCRAFT ENGINE REPAIR SYSTEMS

by

Henry S. Givray and Robert A. Slon

Prepared for Naval Weapons Engineering Support Activity, Weapons Systems Analysis Department (ESA-19), Washington Navy Yard, Washington, D.C. 20374, Task NR 042-335.

School of Operations Research
and Industrial Engineering
Upson Hall
Cornell University
Ithaca, New York 14853

May 28, 1976

Dr. James Matthesen
Naval Weapons Engineering Support Activity
Weapons Systems Analysis Department (ESA-19)
Washington Navy Yard
Washington, D. C. 20374

Dear Dr. Matthesen:

It is our pleasure to submit this report and documentation on the Simulation for the Performance of Aircraft Engine Repair Systems (SPAERS) to you. It was designed, and fully meets the requirements for the degree of Masters of Engineering at the School of Operations Research and Industrial Engineering at Cornell University.

We trust that it will be of service to you and your staff in verifying and comparing Naval aircraft engine repair system configurations.

Sincerely,

Henry S. Givray

Henry S. Givray

Robert A. Slon

Robert A. Slon

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Master of Engineering Report	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): SPAERS: SIMULATION FOR THE PERFORMANCE OF AIRCRAFT ENGINE REPAIR SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED: Research Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Henry Givray, Robert Slon		8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-1172
9. PERFORMING ORGANIZATION NAME AND ADDRESS: School of Operations Research and Ind. Eng., Cornell Univ., Ithaca, NY 14853		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 042-335
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research (code 436), Dept. of the Navy, Arlington, VA 22217		12. REPORT DATE: May 1976
		13. NUMBER OF PAGES: 204
II. CONTROLLING OFFICE NAME AND ADDRESS		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Also supported by the Naval Weapons Engineering Support Activity (ESA-19) - Systems Analysis, Washington Navy Yard, Washington, D.C.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Aircraft Engine
Simulation
Multi-echelon System
Navy Supply System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The allocation of spare aircraft engines is critical to the U.S. Naval aircraft operation's performance. An aircraft in this system becomes inoperative in the event of an engine failure and remains in that state until it is replaced by a serviceable engine. An engine is removed upon failure and subsequently is recovered by repairing it at the location's repair facilities or elsewhere. However, the availability of a spare engine at the location could

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

reduce the amount of time which an aircraft spent in an inoperative state due to engine repair time.

Analytical models have been developed by the DOD to calculate spare engine requirements throughout the system. The models require assumptions be made about certain parts of the system. One important assumption is that carriers are assumed to be continuously connected to the repair system. This assumption ignores the fact that in reality their only connection to the repair system is at a port. The implication is that under this assumption, carriers may swap a failed engine for a serviceable one at any time, directly with the repair location.

In the real situations, carriers perform this trade using the port as an intermediary and only at its time of its docking. The effect is that in reality, ports see more engines in their stock than they do in analytical models. If the port supports a flying activity at that location, the additional engine availability increases its flying performance and may decrease that of the carriers.

A Simulation for the Performance of Aircraft Engine Repair System (SPAERS) was developed to simulate different configurations of an aircraft repair system. The analysis section of this report shows a comparison between the two situations described above, namely the analytical rendering of the repair system, and a situation more closely resembling the real system dynamics.

To a significant level (.05), differences in the performance criterion* were detected between the two configurations of the same system at the port whch supported flying activity, and on one of the more active carriers.

Further study of similar repair systems should be made to support more general statements and observations concerning the relations of carriers and ports to the entire system.

It was shown, however, that SPAERS can be used as a tool for verifying the performance predicted by analytical models for selected repair systems, and enables thier subsequent statistical comparison.

---

*Criterion is defined as: average down aircraft at a base divided by number of aircraft at that base. A down aircraft is defined as an aircraft unable to 'fly' due to the failure of one of its engines.

SPAERS

Simulation for the Performance of Aircraft Engine Repair Systems

Henry S. Givray
Robert A. Slon
Masters of Engineering Project Report
Cornell University, School of Operations Research & Industrial Engineering
May 28, 1976

prepared
for

Naval Weapons Engineering Support Activity
Weapons Systems Analysis Department (ESA-19)
Washington Navy Yard
Washington, D.C. 20374

PREFACE

In any system where the availability of physical components determines the operational quality of the system, inventory management plays a key role. The United States Navy's aircraft operation is an example of this situation. The physical components in this case are aircraft engines, and the operational quality of the system is measured by aircraft readiness. Here the components fail over time and can be subsequently recovered through the repair system. A failed engine renders an aircraft inoperative until a serviceable engine becomes available as a replacement. When a demand for an engine is not met, it is said to be backordered and met at a later date. To compensate for the time associated with repairing an engine, spare engines are pre-positioned at various locations. These locations which support repair, supply, and flying activity are called bases.

The Department of Defense presently uses a method, described in DODI 4230.4 [1], to establish spare engine stock levels for each base. The method's objective is to stock each base so that the base has outstanding backorders for only a small proportion of time.

The DODI 4230.4 method of establishing spare stock levels at a particular base does not consider an important

system interaction. In particular, it does not account for the manner in which resupply times may vary with the spare stock levels at resupply bases.

Consequently, a model called NAVMET [2] was developed which explicitly considers the effect of stocking inventory at one location on the availability of engines at other locations. NAVMET's objective is to optimize the quantity and allocation of spare engine with respect to a constraint on average down aircraft. It is applied in the environment of the Navy's multi-echelon repair system.

The dynamic nature of the Navy's repair system, particularly the aircraft carriers' relation to the system have raised some questions concerning both models' assumptions about the demand process. These questions, which will be developed more fully in this report, formed the basis for a Masters of Engineering at Cornell University. The project involves modelling the dynamics of the actual system by means of a digital computer simulation. The simulation will be used to study the effect of an assumption made in the analytic models regarding carrier operations on expected system performance.

This simulation, called SPAERS (Simulation for the Performance of Aircraft Engine Repair Systems), was developed for the Naval Weapons Engineering Support Activity (ESA-19). It was designed as a tool for observing the long run performance of a repair system given its system parameters.

SPAERS input may be adjusted to simulate the environment in either the configuration assumed by the analytical models, or the one which more closely resembles the real system. This will enable a comparison of the system's performance to be made between the two modes of operation, given the same spare engine stock levels.

The authors wish to thank the faculty and members of the School of Operations Research and Industrial Engineering at Cornell University for their valuable consultation. In particular: Professors John Muckstadt, William Maxwell and Thomas Santner; and Dr. James Matthesen and Mr. William Booth of the Naval Weapons Engineering Support Activity.

H.S.G. and R.A.S.

ABSTRACT

The allocation of spare aircraft engines is critical to the U.S. Naval aircraft operation's performance. An aircraft in this system becomes inoperative in the event of an engine failure and remains in that state until it is replaced by a serviceable engine. An engine is removed upon failure and subsequently is recovered by repairing it at the location's repair facilities or elsewhere. However, the availability of a spare engine at the location could reduce the amount of time which an aircraft spent in an inoperative state due to engine repair time.

Analytical models have been developed by the DOD to calculate spare engine requirements throughout the system. The models require assumptions be made about certain parts of the system. One important assumption is that carriers are assumed to be continuously connected to the repair system. This assumption ignores the fact that in reality their only connection to the repair system is at a port. The implication is that under this assumption, carriers may swap a failed engine for a serviceable one at any time, directly with the repair location.

In the real situations, carriers perform this trade using the port as an intermediary and only at its time of

its docking. The effect is that in reality, ports see more engines in their stock than they do in analytical models. If the port supports a flying activity at that location, the additional engine availability increases its flying performance and may decrease that of the carriers.

A Simulation for the Performance of Aircraft Engine Repair System (SPAERS) was developed to simulate different configurations of an aircraft repair system. The analysis section of this report shows a comparison between the two situations described above, namely the analytical rendering of the repair system, and a situation more closely resembling the real system dynamics.

To a significant level (.05), differences in the performance criterion* were detected between the two configurations of the same system at the port which supported flying activity, and on one of the more active carriers.

Further study of similar repair systems should be made to support more general statements and observations concerning the relations of carriers and ports to the entire system.

It was shown, however, that SPAERS can be used as a tool for verifying the performance predicted by analytical

---

*Criterion is defined as: average down aircraft at a base divided by number of aircraft at that base. A down aircraft is defined as an aircraft unable to 'fly' due to the failure of one of its engines.

models for selected repair systems, and enables their subsequent statistical comparison.

TABLE OF CONTENTS

I

INTRODUCTION

United States Naval aircraft operations depend on the quantity and distribution of spare aircraft engines throughout its aircraft engine repair systems. As aircraft fail over time, the failed engines are subsequently recovered by the repair system. Aircraft down time due to repair system lead times can be reduced by allocating spare engines to particular system locations (bases). An engine failure is considered a demand upon the system which can only be satisfied by a serviceable engine. If a demand cannot be satisfied immediately, it is backordered and filled as soon as possible.

One method of spare engine allocation presently used by the Navy is described in the Department of Defense Instruction 4230.4 [1]. Its objective is to stock each base for a small proportion of time. This method, however, ignores the effect of the spare engine supply levels on resupply times.

NAVMET [2] takes stock dependent resupply times into consideration. Its objective is to maintain the expected number of backorders to flying activity at specified minimal levels. NAVMET produces an optimal solution to the problem

of allocating quantities of spare engines within the Navy's four echelon repair systems with respect to a constraint on average time weighted downed aircraft.

The system dynamics, particularly those involving aircraft carriers, present some reasonable doubt as to the validity of the demand process that is assumed for carriers in both of these models.

The following report presents the development of a computer simulation model, called SPAERS (Simulation for the Performance of Aircraft Engine Repair Systems), which serves to statistically compare the actual operation of the repair system with the manner in which it is assumed to operate by the analytical models. The purpose of this report is to present the system characteristics which are relevant to model formulation, a discussion of the analytical methods for establishing stock levels in the system, and the motivation for a computer simulation of the system. As SPAERS will be used by the Naval Weapons Engineering Support Activity, as a selective verification procedure for predicted system performance, the manner in which the simulation treats the real system is discussed in detail. This discussion includes the simulation's initial conditions, model assumptions, and user availabilities.

Analysis includes the results of a specific system simulation. Stock levels were computed for the system using the DODI calculation, and two runs were performed. One

operated the system in the manner which is assumed by the model, and the second run operated the system using the actual system dynamics. Finally, some observations were made concerning the role of the port in computation of stock levels.

II

RELEVANT SYSTEM CHARACTERISTICS

In order to aid the discussion of the repair system in later sections of this report, some features of the aircraft engine repair environment which are important and often assumed by a mathematical formulation will be described.

A base is defined as a location where supply, repair, or flying activity occurs. The base's flying activity generates failed engines which are sent to the repair function. If the repair function at this base does not have the facilities to perform the repair required, the engine is sent to another repair base in exchange for a serviceable (operational) engine.

Engines which complete repair at a base become part of the base's serviceable stock supply. The supply function is responsible for maintaining the flying activity at its own base. In addition, it must exchange a serviceable engine for a failure which is sent to its own location for repair (see figure 1).

The base repair facility's ability to deal with failures depends on the sophistication of its repair equipment. A base is identified by the most complex repair

BASE ACTIVITY
(engine flows)

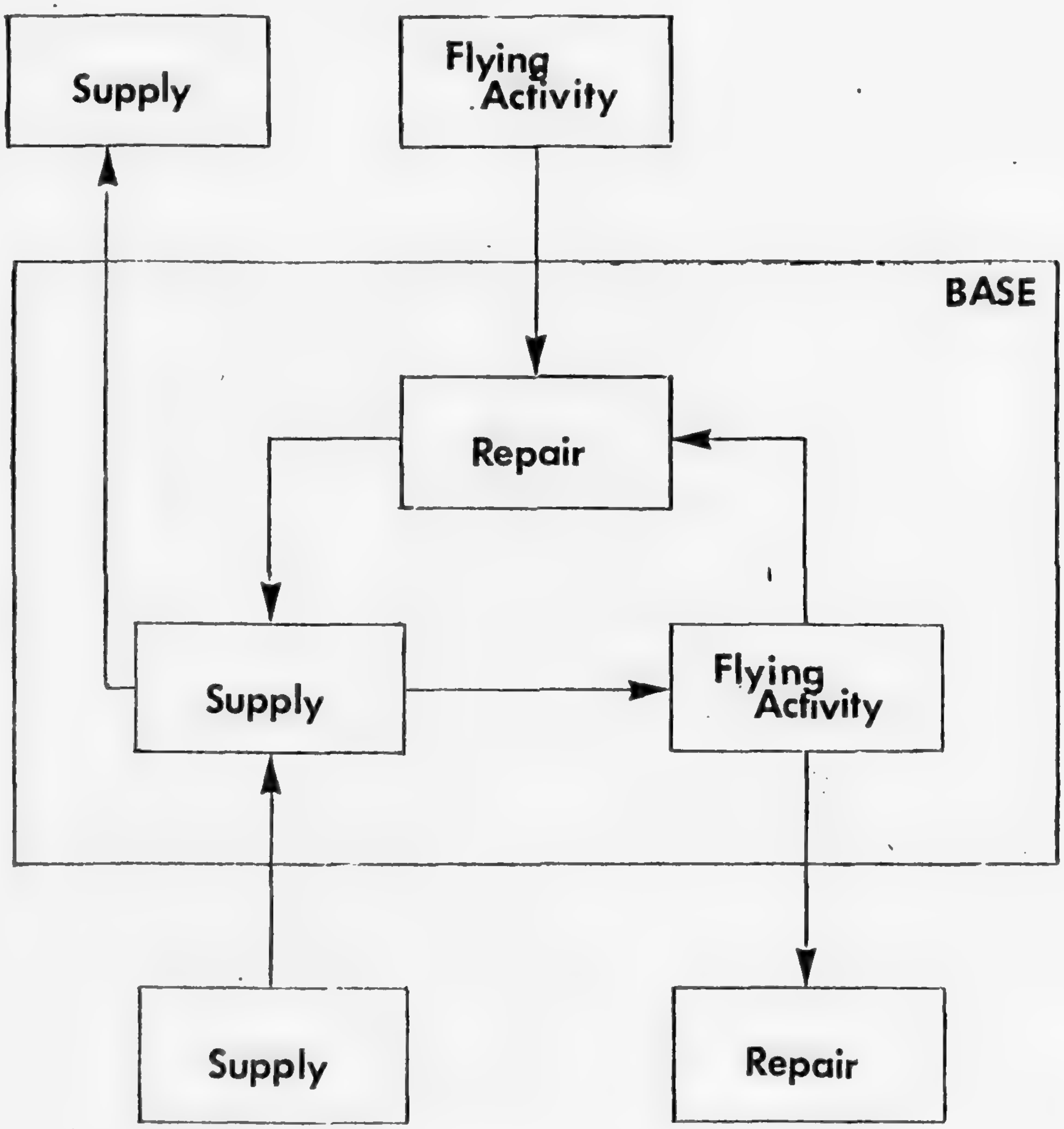


Figure 1

that it can accomplish. Five degrees describe engine repair:

4° no repair
3° minor repair
2° moderately complex repair
1° major repair
0° overhaul*; major repair

If an engine fails at a base which cannot perform the required degree of repair, the engine is sent to a higher echelon (lower degree) repair base. In this case the flying activity at which the failure occurred demands an engine from the supply function at its base. The fact that an engine was sent for repair to another base requires that the repair base return a serviceable engine to the supply at the base which experienced the failure (see figure 2). The repair system is fixed in the sense that all j° failures which occur at base I are always repaired at a base K for all I and j**.

A supply function's inability to meet any demands is registered as a backorder and filled as soon as possible.

A repair system (see figure 3) is defined as a set of bases having a common 0° repair base for a particular engine model type. The types of bases which comprise a system are described as follows.

---

*Periodically, engines are removed from aircraft for scheduled overhaul even though they have not failed.

**Occasionally, in the real system, a 2° failure occurring at a 2° repair base will be sent to a 1° repair base. A similar situation occurs with 1° failures and overhaul locations.

DEMANDS

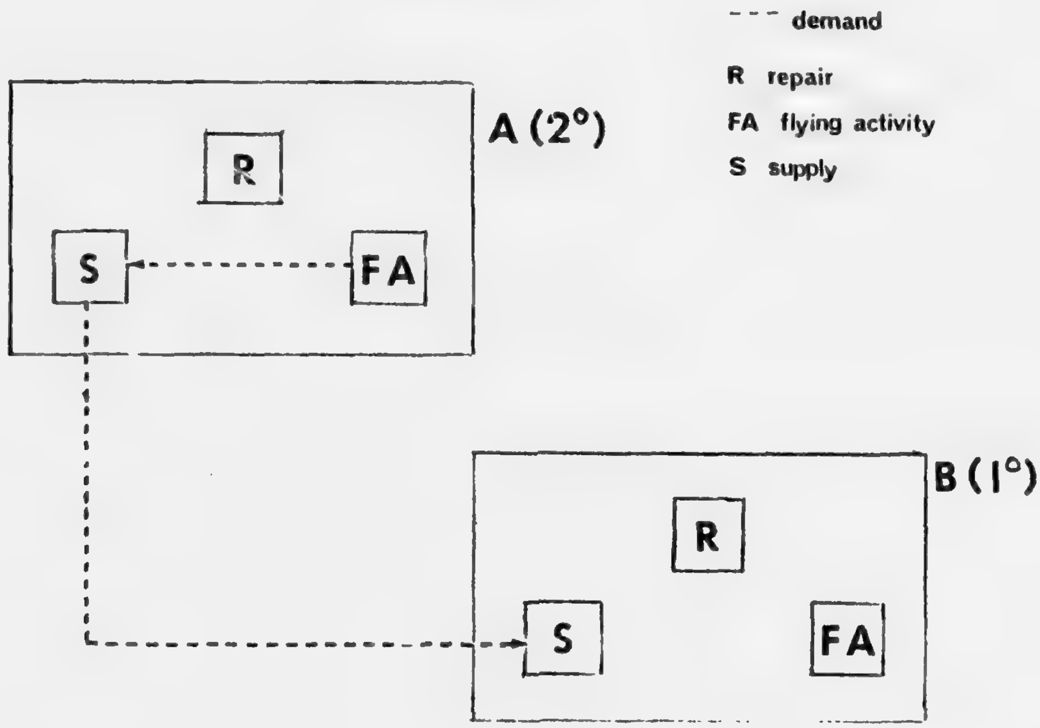


Figure 2

In the above figure, base A is a 2° repair base and B is a 1° base. Suppose the flying activity at A has experienced a 1° failure. Thus A's repair facility cannot repair the engine, and it must be sent to B. The two demands which occur are:
1) Flying activity on Supply Function at A, and
2) Supply Function at A on Supply Function at B.

1) A carrier has its own "on board" supply function and flying activity. When a carrier is deployed, its link with higher echelon repair is broken and its flying activity performs. A non-deployed carrier is connected to the repair system, but its flying is inactive.

Due to the carrier's particular relationship with the rest of the repair system, on-board engine failures which the carrier is unable to repair will not enter the repair system until the carrier becomes non-deployed (docked)*.

During the dock time, the port sends the carrier's failed engines to their respective repair bases in exchange for serviceable engines. If some carrier demands are not satisfied durings its docking, they are backordered at the port.

2) A port has full responsibility for supplying carriers before they are deployed. A port may have any number of carriers to service, but a carrier is always serviced by one port. The port may also support its own flying activity.

3) The depot performs overhauls (0° repair) for the system and does not generally support its own flying activity.

The "pipeline" describes the connections between bases or intra-base functions and is measured in time durations. Engines may be said to flow through the pipeline

---

*In special cases in the acutal system, a serviceable engine may be delivered to a carrier while it is deployed.

at a constant rate.

A "twisted pipeline" is used to explain the connection between carriers and port. When the carrier is deployed, its pipeline to the port is cut off, or in a sense, "twisted" so that the flow of engines may not reach the port until the carrier docks. It may be noted here that present models for determining spare engine requirements at a base ignore the twisted pipeline configuration. Rather, they assume a fixed, continuous resupply time for a carrier (as if it were a ground base). The term, "continuous pipeline" will be used to designate the treatment of carrier activity assumed by the analytical models. (The motivation for this study is the validation of the continuous pipeline assumptions.)

Resupply time is defined to be the average amount of time which it takes for a failed engine to be replaced by a serviceable one at each base.

A base is said to have custody of an engine, in the sense of ownership, if the engine is en route to the base, or in the base's repair or supply activity. Only serviceable engines may be used to satisfy a demand.

REPAIR SYSTEM

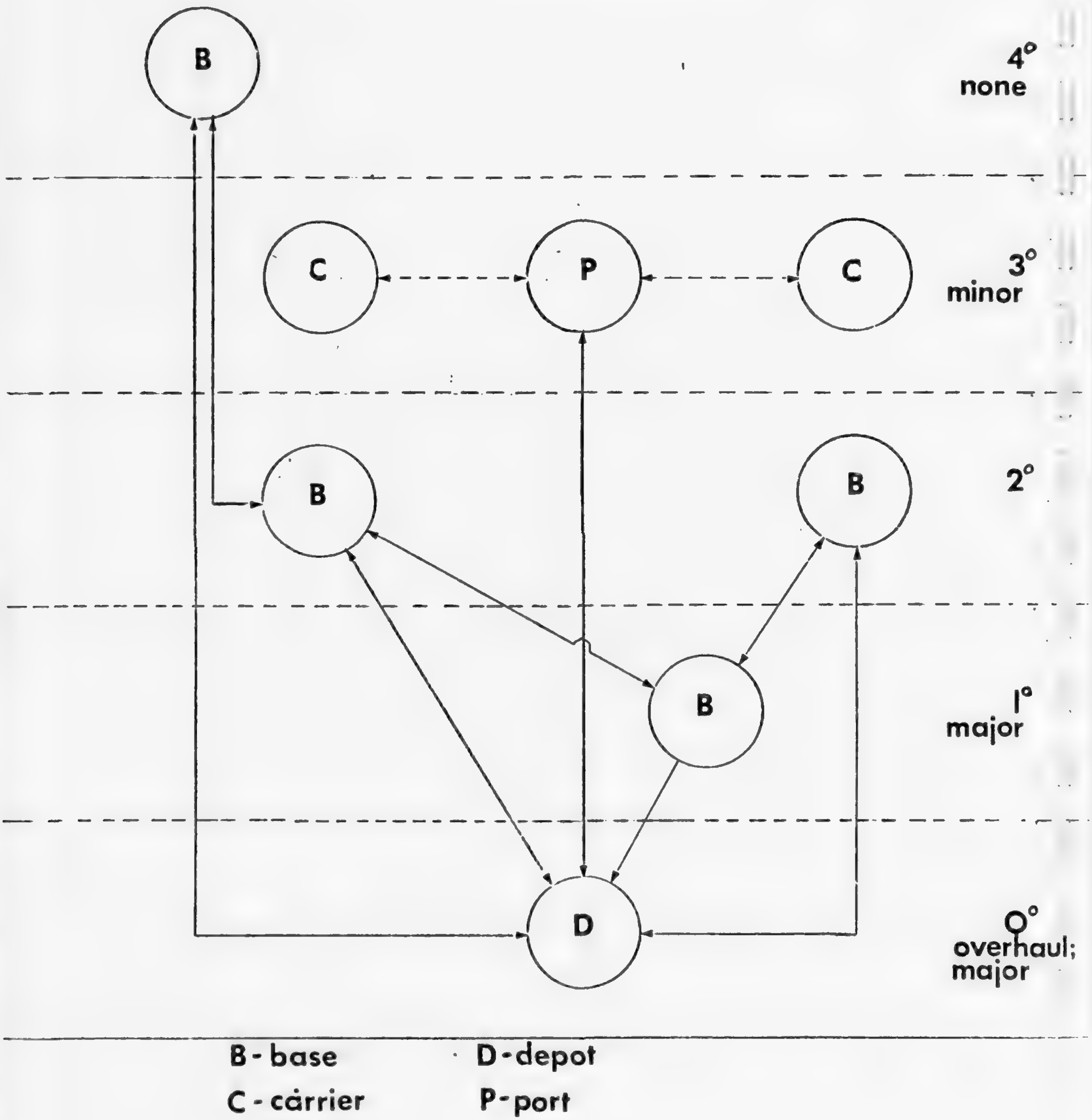


Figure 3

III

SYSTEM PARAMETERS

The previous section described system activity which is relevant to mathematical formulation. This section will explain the parameters which describe the repair system operation.

One engine model type is a necessary constraint in defining a system. Relevant flying activity, then, consists of those aircraft types having the system engine. These aircraft are considered a base's permanent property.

The combination code describes the aircraft type's use of the system engine. It is associated with failure rates and the probability of j° failure for the system engine where $j=0, 1, 2, 3$.

The various rates, probabilities and times which are used as parameters for the system can be found in computer output from the U. S. Navy's PASER*. The system "geography" is described by a matrix $\underline{N}$ where n_{ij} is the base number to which a $\underline{j}$ degree failure is sent given that it occurs at base $\underline{i}$. $\underline{N}$ is easily derived from the PASER data.

---

*Projection of Aircraft Spare Engine Requirements is a set of computer software used by the U. S. Navy for computing Spare Engine Requirements.

A specific flying rate (in hours per month) is assigned to aircraft types at a base. Any number of aircraft types and numbers of that aircraft type may be found at a base. A daily flying rate is assumed to be 1/30 of the flying hours per month.

Travel time from base I to base J is assumed to be constant as are repair times for a given failure degree regardless of where that repair takes place.

The discrete probability distributions describing the probability of a $j°$ failure are available from the same PASER computer records as are mean times between removal for a given combination code.

Other system inputs include the number of engines per aircraft (UPA) and the desired performance criterion. The criterion will be defined by the average time weighted downed aircraft which are due to engine failures divided by the total number of aircraft which utilize the system engine at that base.

IV

BACKGROUND

The system described in the previous section operates for the purpose of maintaining flying activity. Measures which indicate the system's effectiveness are in number of down aircraft at any point in time and in the durations of inoperative aircraft. Other measures of effectiveness would be described by a supply function's ability to meet demand, and the numbers and durations of its backorders.

An obvious limitation on the performance of a supply system is the number of spare engines that are in the system to compensate for repair and travel times which are involved in recovering engines. At the same time, it is equally important that spare engines are located at the "correct" supply function in order to achieve acceptable levels of flying performance.

This section will describe some of the performance measures and explain two models that are used by the Department of Defense to allocate the appropriate number of spare engines in an attempt to optimize using these criterion measures. Furthermore, the need for a simulation model to statistically verify some of the assumptions made by the DOD

models will be demonstrated.

Measures of Performance

The ready rate may be defined by the probability that there are less than S units in resupply at a base where S is a decision variable.

$$RR = \sum_{x=0}^{S} p(x|\lambda T),$$

where $p(x|\lambda T)$ is the probability of having x units in resupply.

The major assumption here is that demands follow a Poisson distribution with rate λT, where λ = daily failure rate and T = average resupply time. Resupply at a base is defined to be physical, on-hand engines plus the number of engines in base repair plus serviceable stock from other bases, less backorders.

Fill rate is the proportion of demands that can be filled immediately, or the probability that S - 1 units are in resupply

$$FR = \sum_{x=0}^{S-1} p(x|\lambda T)$$

Again the demand process is assumed to be Poisson.

Although these measures are used extensively by the DOD to compute requirements, they do not reflect system performance in a complete sense. Neither the number nor the duration of backorders can be inferred from these measurements.

A more suitable measurement for both planning and

evaluation purposes is a base's time-weighted average backorders, measured in backorder-days per day. A mathematical rendering of this measurement is given by:

$$B(S) = \sum_{x=S+1}^{\infty} (x-S)p(x|\lambda T)$$

where S is the on-hand stock at the repair base. The measure has relevance only at bases with flying activity. Therefore, a non-flying base would not calculate a backorder criterion. Its stock, however, effects the performance of the flying bases it resupplies.

This criterion can be normalized over all bases by dividing it by the total aircraft at the base. Note that a convenient aspect of this measurement is its convexity in S, given T, making it a reasonable objective in optimization models.

Spare Stock Level Determination

Given the above performance measures and the Poisson demand assumption, the Department of Defense rationale for supplying safety stock can be illustrated. First, the parameters λ and T are approximated. Daily Demand Rate (λ) for engines at a base can be approximated by multiplying average flying hours per month by the number of aircraft at the base, by the failure rate per engine times the average number of engines per aircraft at the base. This quantity is in terms of engine failures per day. The average resupply

time $\underline{T}$ is derived by multiplying the probability of a j° failure at a base by the $\underline{j}$ travel time to the j° repair base and summing over all $\underline{j}$ plus the repair time at base $\underline{A}$ times the probability of repair at base $\underline{A}$ (see figure 4). By using the desired ready rate criterion of α, the safety stock level $\underline{S}$ can be calculated for the base $\underline{A}$ by solving

$$\sum_{x=0}^{S_A} p(x|\lambda_A T_A) \geq \alpha$$

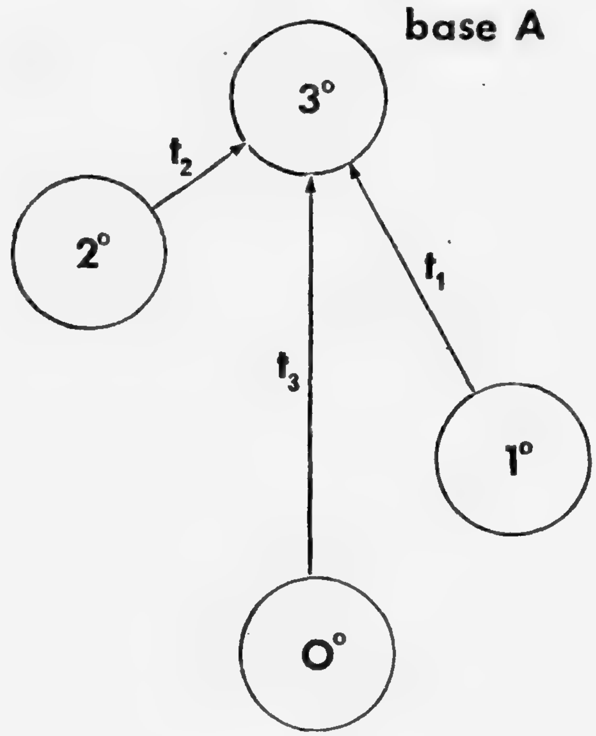
This, in fact, is the method used by the Department of Defense.

It becomes clear, that for any value of α, $\underline{S}$ is fixed, and the other performance measures are implicitly determined.

The difficulty with determining an $\underline{S}$ (spare stock level) using the method above lies in the fact that the average resupply time $\underline{T}$ is not truly represented. Clearly, if the j° repair base had some on-hand stock, resupply time from a j° repair base could be shortened by the time it takes to repair an engine. That is, instead of incurring a delay due to repair time, an engine could be sent off immediately. In a probabilistic sense, this delay would vary with the amount of spare stock which the repair base had. T_A might now be represented as

$$T_A = \sum_{j=0}^{3} p_j(t_j + H(S_j))$$

where S_j is the amount of safety stock at base $\underline{j}$. $H(S_j)$ can



$$T_A = \sum_i p_i(t_i) + p_3 r_3$$

$p_j = p\{ j^{\circ} \text{ failure}\}$

r_j = repair time for a j° failure

t_j = travel time to base A from j° repair base

Figure 4

be defined as the number of backorder days per demand such that,

$$H(S_j) = \frac{1}{\lambda} \sum_{x=S+1}^{\infty} (x - S_j)p(x|\lambda T_j)$$

where $\lambda = \sum p_j\lambda_j$ or the system demand on the repair base.

Thus, it can be seen that system interdependence is considered by describing the resupply time as a function of the repair base stock. The NAVMET [2] model employs the above function and develops an algorithm for determining the optimal quantity and distribution of spare aircraft engines.

The application of this model to a four echelon system involves a dynamic approach whereby subproblems which segregate bases into j° "families" are solved. Full exposition of the algorithm is beyond the scope of this report. The interested reader is referred to Naval Publication R-7511 (NAVMET)[2].

The optimality of a solution obtained from the NAVMET algorithm requires the validation of some assumptions about the system. One which requires discussion at this point, and serves as the motivation for this investigation concerns the continuous pipeline.

Recall that in the actual system, aircraft carriers leave port, and for all practical purposes, are not in contact with the repair system for the duration of their deployment. In this way, the repair pipeline is paid to be twisted.

The NAVMET and the DODI 4230.4 approach, both assume

that carriers resemble ground bases and are always connected to the repair system with a fixed resupply time. Carrier resupply times are approximated by the average resupply time which they would experience in the actual, twisted pipeline situation.

Clearly, the surges and lags which the twisted pipeline carriers impose upon the repair system cast aspersious on the ability of the Poisson process to describe the demand process. Since the description of the demand process is crucial to finding an optimal solution to the allocation problem, it is desirable to note the conditions under which the continuous pipeline is a valid approximation to the twisted pipeline.

The question, then, is one of degree. On the basis of performance measures, especially the backorder criterion, how closely does the continuous pipeline configuration resemble the twisted pipeline? Other questions might be posed: What parts of a repair system feel the effect of a twisted pipeline? If any, what relationships do carriers' continuous pipeline performance have with its demand rate and deployment schedules in the twisted pipeline?

The dynamics of a twisted pipeline renders it too complex to model analytically. A digital computer simulation seemed to provide the most convenient approach to an effective comparison. SPAERS or the Simulation for the Performance of Aircraft Engine Repair Systems was developed

to statistically compare performance of a twisted pipeline configuration with that of the continuous. In addition, the NAVMET performance predictions could also be compared with those of SPAERS.

V

SPAERS

In the previous section the present methodology for determining spare aircraft engines was discussed. It was pointed out that a procedure that would test the validity of the analytical models was needed. Furthermore, it was determined that a simulation approach could adequately accomplish this task.

The SPAERS model is designed to examine the performance of the "optimal" spare stock levels determined by the anaylitical model, NAVMET, under dynamic conditions.

In this section the SPAERS model is discussed. First the underlying assumptions and initial conditions of the model are outlined. SPAERS options and capabilities along with SPAERS data output are briefly described next. The model's breakdown of the real-world system in terms of entities (objects of the system), their attributes (characteristics of the entities) and events (changes in the state of a system entity) is provided at the end of the section.

A detailed description of the construction and functioning characteristics of the model may be found in Appendix A. User instructions for operation are provided in Appendix B.

A. Assumptions

Some assumptions were made in SPAERS regarding the following entities:

Engines/Aircraft

SPAERS assumes that engines are not removed from one inoperative aircraft to put another aircraft up (no cannibalization). Only serviceable engines removed from stock may be used in this capacity. Nor is any distinction made between engines in stock in terms of their life or age. The choice of which engine services which aircraft is totally random in that a failure time is generated only after installation. Engines which are servicing aircraft with same combination codes will deteriorate at the same rate (flying hours between removals). Likewise, two engines on the same aircraft will deteriorate at the same rate.

The engine failure (demand) process is assumed to be a Poisson process, implying exponential failure times.* A Poisson process also supports other basic assumptions, namely, that engines will be found uniformly distributed throughout system pipelines. Secondly, the memoryless property of the exponential distribution allows failure times to be generated according to the same distribution regardless of the engine's past experience.

*NAVMET argues against Poisson demands because of the uncertainty in predicting the demand rate.

Repair times and travel times between bases are assumed to be constant. This serves to reduce the system variance and improve the statistical accuracy of the performance measures.

Average repair times were interpreted to include the waiting time which an engine would experience in the repair function. For this reason, the repair function is assumed to have an infinite number of repair facilities and processing times will include average waiting times.

The discipline for filling backorders at a base is first in first out. At bases which are not ports, no engine may be in stock while a backorder exists at that base.

Aircraft flying rates are given in flying hours per month for a given aircraft at a given base. In determining a failure date, SPAERS assumes this flying rate to be uniformly distributed throughout the month.

Carriers

The repair system is described as a tree in which all failures of type $\underline{j}$ occuring at base $\underline{i}$ are always repaired at the same base. In the case of a port which supports no flying or repair activity, the same relationship holds as if the carrier failures occurred at the port. That is to say, that j^{th} degree failures which are delivered to the port from any carrier are sent to the same repair base.

The fact that a carrier's flying activity is only active during deployment supports the following assumptions.

1) No failures can occur during a carrier's port time.

2) Statistical observations are made on a carrier only during deployment. (Backorders or other system conditions do not affect carrier's performance during its docking.)

Although a carrier's flying activity is dormant in port, its repair function is not. It is not unlikely, then for an engine to become available on board a carrier through the efforts of the on-board repair function. All other resupply, however, is accomplished by the port concurrent with the time of deployment. No resupply, external to that of the carrier's own, can be performed during its deployment.

Engines on board a docked carrier cannot be used by the port to supply an out going carrier, nor can it use those engines for its own flying activity.

SPAERS incorporates some variability in a carrier's port/deployment schedule by using the normal distribution. It was believed that a fixed, non-varying carrier schedule would make the demand process too regular. No information is readily available on carrier schedules except, perhaps, for <u>average</u> port or deployment times. So the intuitive appeal in using the normal distribution to vary the schedule lies in the ability to control the mean and variance according to the information available, without introducing any special assumptions about the particulars of the scheduling activity.*

---

*The ability to use fixed schedules is available. (See Appendix B on user availability.)

SPAERS event scheduling introduces some assumptions concerning the repair system in the chance that two events may occur simultaneously in the simulation. The event priority is as follows:

1) Engine becomes available.
2) An aircraft fails.
3) A carrier will dock or deploy.
4) A sampling event occurs.

Engine availability is executed first to avoid the unnecessary deactivation of an aircraft or the deployment of an under-stocked carrier.

An aircraft is allowed to fail before a carrier docks to avoid the event that an engine fails on board a docked carrier.

The sampling activity is considered last so that all pending events may occur before the system statistics are calculated.

Simultaneity of events is a rare event, both in SPAERS and in reality, but this priority rationale prevents any difficulty in executing the simulation events.

B. Initial Conditions

As in many simulation studies, SPAERS is concerned with investigating the 'long run' or 'equilibrium' behavior of the system. Unfortunately, a simulation model must be started and stopped. Because of the factitiousness introduced

by the abrupt beginning of a simulation run, the performance of the simulated system does not become representative of the corresponding real-world system until it has essentially reached a "steady-state"* condition. Thus, we must be concerned with the problem of how to obtain data that are relevant for predicting the 'steady-state' behavior of the real system. In other words, we would like to observe data from a system operating under 'chance causes'--random variation, and not a system whose underlying probabilistic behavior is dependent on initial conditions.

There are two traditional ways of dealing with the 'initial conditions' problem:

- Run the simulation model for some time without collecting data until the simulated system has "essentially" reached a state representative of steady-state conditions. This time is usually referred to as a 'run-in' or 'stabilization' period.
- Start system in a state as representative of steady-state conditions as possible to minimize the required length of the 'run-in' period.

It is difficult to estimate how long this 'run-in' period needs to be. Furthermore, some data are not available to fully start the system in a state representative of

---

*The term "steady-state" has the probability theory definition of describing the state of a system as having reached a limiting equilibrium probabilistic behavior, independent of time.

steady-state conditions.

The authors have included in SPAERS a hueristic method for calculating a 'run-in' period. Also, some knowledge is known about the system to accurately initialize portions of the system state sufficiently close to steady-state conditions. They are discussed below.

Initial States of Certain Entities:

- Aircraft #i, i = 1, 2, • • •, total number of aircraft. Since no data are available on how many aircraft on the average are down at any particular base, all aircraft are initialized as being "operative." This means the down aircraft queue is empty and the failtime queue is full. Thus, initially SPAERS calculates the life of each engine on aircraft i. The previous history of an engine need not be considered since the exponential distribution has the memoryless property.

  A failure time must also be computed for aircraft i.
- Base #i, i = 1, 2, • • •, total number of bases.

  Since all aircraft are initialized as being in an 'operative' state, the number of backorders and the number of down aircraft at base i are both equal to zero. Furthermore, the stock level at base i equals the number of spare engines belonging to base i.

  There are two components of base i stock:

  (1) Safety stock

(2) Pipeline stock

Both of these quantities are either computed in the initialization using the 'DOD requirements' method or are data inputs (user option--see Appendix B). The on-hand stock level of base i is initialized to equal the safety stock. Each engine of pipeline stock, on the other hand, is initialized uniformly in base i's pipeline. Thus, by scheduling the availability of these engines to base i in the future, we are closer to attaining steady-state conditions; that is, in the real system you expect to observe a certain number of engines in base i's pipeline (resupply) and rarely will all engines exist as on-hand stock.

- Carrier #1, i = 1, 2, · · ·, total number of carriers

The state of carrier i (deployment or docked) is randomly determined with Probability {STATE(i) = 'AT SEA'} = mean time at sea/(mean time at sea + mean time at port) and Probability {STATE(i) = 'AT PORT'} = 1 - Probability {STATE(i) = 'AT SEA'}.

The time of next state change of carrier i is initialized at a fraction of its first schedule time. This schedule time may be deterministic or probabilistic (user option--see Appendix B).

Thus,

Time of next deployment = U · first deployment time

if STATE(i) = 'AT PORT'

or

$$\text{Time of next docking} = U \cdot \text{first docking time},$$
$$\text{if STATE(i) = 'AT SEA'}$$

where $\underline{U}$ is (0, 1) uniformly distributed random variable.

RUN-In period:

At any point in time an engine may exist as on-hand stock, pipeline stock or as an operating unit on an aircraft. The spare engines have appropriately been classified and located in the system accordingly as an attempt to achieve steady-state conditions. Due to the lack of information, we were forced to initialize the remaining engines in the system as 'operating units on aircrafts.' To alleviate this artificial condition, it seems intuitively appealing to allow all these engines to fail at least once before any collection of system data. By allowing all engines in the system to exist as on-hand or pipeline stock or as operating units, the system has come one step closer to achieving steady-state conditions.

Specifically, the run-in period is computed by:

(1) Calculating the average time it takes for all 'operating' engines to fail at base $\underline{i}$.

(2) Obtaining the maximum time over all 'flying' bases.

(3) Advancing the calculation to the nearest whole day.

It should be noted that during SPAERS initialization a report of system parameters and spare stock levels is printed.

C. SPAERS Options and Availabilities

Variations of system parameters accomplished through data input provide the user the opportunity to study a system's performance and sensitivity under different conditions.

Some examples are:

- Twisted vs. Continuous Pipeline Configurations
- Probabilistic vs. Deterministic Carrier Deployment and Dock Times
- If probabilistic carrier times are employed, then the effect of high variance or instability may be studied.
- The effects of over or under stocking various bases may be examined.
- DODI-4230.4 [1] calculated spare stock levels with some confidence level vs. user data input spare stock levels.
- Effects of including (excluding) a port which maintains neither a flying nor a maintenance activity.

For a detailed discussion on these and other SPAERS availabilities the reader should refer to Appendix B.

D. Data Output

At the termination of the simulation, SPAERS reports statistical measures which summarize base performance activity during the simulation run.

Measures such as the ready rate, fill rate, average on-hand stock, average backorders, average down aircraft, average backorder time, average down aircraft time along with non-zero distribution values of net inventory (on-hand stock—backorders) and down aircraft are printed for each base. Furthermore, the sample mean and standard deviation of the mean along with a confidence interval of the criterion (average down aircraft/total number of aircraft) for each base are provided.

Finally, two measures of system criterion are reported to the user. A detailed discussion of the meaning and implications of these measures is provided in Appendix C.

E. SPAERS System Breakdown

Entities and their Attributes

Entity:	Attribute:
(1) System--	
	• Engine Model Type
	• Total number of bases
	• Total number of carriers
	• Total number of aircraft
	• Total number of aircraft types

Entity:	Attribute:
(ii) Engine Model Type--	
	• Name
	• Total number of 'combination codes'
	• Repair time for each degree of failure
(iii) Base--	
	• Name
	• Assigned 'base' number
	• Assigned 'carrier' number ('0' if ground base)
	• On-hand stock level at any point in time
	• Number of backorders at any point in time
	• Total number of backorders up to any point in time
	• Number of down aircraft at any point in time
	• Total number of down aircraft up to any point in time
	• Number of aircraft types
	• Number of aircraft of each type
	• Total number of aircraft
	• Total number of engines

Entity:	Attribute:
	• Travel time to every base in system
	• Base at which an engine will be repaired when failure degree $\underline{i}$ occurs. $(-1 \leq i \leq 3)$
	• Desired criterion
	• Weight of importance
	• Highest repair degree
(iv) Carrier--	
	• Time of next deployment or docking
	• State (deployment or docking) at any point in time
	• Port at which carrier docks
	• Future Deployment and Docking Schedules
	• Number of engines that have been sent off to be repaired but have not been satisfied
	• Number of demands (engines) that were not satisfied by carrier's port during last 'docking'
(v) Aircraft--	
	• Assigned number

Entity:	Attribute:
	• Base to which aircraft belongs
	• Type of aircraft
	• Next failure time
	• Remaining life time of engine(s) on aircraft at time of failure (in terms of flying hours)
	• Flying Activity in terms of flying hours per month
(vi) Type of Aircraft--	
	• Name
	• Assigned number
	• Combination code
	• Engine capacity
(vii) Combination Code--	
	• Actual code number used by NAVY
	• Assigned number
	• Probability of $\underline{i}^{th}$ degree failure for all $\underline{i}$ $(-1 \leq i \leq 3)$
	• Mean time between removal (in terms of flying hours)

Events and their Scheduling

An endogenous event takes place as a result of its being scheduled during the course of the simulation

run.* SPAERS endogenous events are as follows:

- The time at which an engine failure causes an aircraft to become inoperative. This event is appropriately referred to as PLANEFAILEVENT.
- The time at which an engine in a particular pipeline becomes available as on-hand serviceable stock at a base. This event is referred to as ENGAVAILEVENT.
- The time at which the state of a particular carrier changes. This event is referred to as SHIPEVENT.

An exogenous event is made to occur by specifying the type of event and simulated time at which it is to occur through data input prior to the start of a simulation run.* This type of event usually serves as a means of controlling a desired aspect of the simulation run. SPAERS exogenous events are as follows:

- The time at which appropriate statistics are to be gathered for analysis purposes (see Appendix AIII for discussion). This event is referred to as SAMPLE.
- The time at which the simulation run is to be terminated and final statistics are reported. This event is appropriately labeled REPORT.

SPAERS synchronization, that is, the identification of the event occurring next and its time of occurrence, is based on asynchronous timing--the search for a minimum among event times (see Flowchart in Appendix AII).

---

*Reference [6], pp. 232-233.

VI

ANALYSIS

To illustrate SPAERS' treatment of the two pipeline configurations, an actual repair system was analyzed. The U. S. Navy's PASER data were used to calculate DODI spare stock levels. PASER also provided system parameters that served as SPAERS input.

AIRPAC Command's Engine type model T58GE-series 8B and 8F in the North Island Naval Aircraft Rework Facility System (Combination Codes 731, 733 and 735) defined the repair system which was simulated by SPAERS in this analysis.

The DODI calculation was performed using the following consideration. According to PASER, ports which do not fly or repair will not enter into spare stock level computations. That is, carriers are connected directly to their repair echelons, ignoring the port. The implication here is that the port is not equipped to satisfy carrier demands, but the carrier repair echelons are.

A 0.9 ready rate criterion was used for all bases. SPAERS showed that the configuration assumed by DODI and stocked accordingly, did, in fact produce ready rates close to 0.9.

To demonstrate the effect of introducing the port to a continuous pipeline situation, the following configurations were run by SPAERS.

1) Continuous carrier pipelines without ports (DOD spares calculation included)

2) Continuous carrier pipelines with ports

3) Twisted carrier pipelines with ports

Comparisons of the down aircraft criterion were made in each case using the test of the hypothesis that the means of two Normal Distributions are equal, assuming that the standard deviations are not known and not necessarily equal* (.05 level). The test was performed between corresponding bases in two different configurations.

The first comparison involved 1) and 2) to show any significant effect on performance by the introduction of ports to the original DODI system.

As the results in table 1 indicate, only Cubi Point, a port, shows a significant difference in performance. In reality, Cubi Point repairs and supports its own flying activity. By virtue of its port activity, it experiences higher engine availability by storing engines for carriers. But these engines may be used for Cubi Point's own flying activity and serve to decrease its down aircraft time.

Another comparison was made between configuration

---

*Reference [5], pp. 240-242.

2) and 3). Here the singular effect of a twisted pipeline is contrasted with the continuous pipeline. Again, Cubi Point is significantly different from the continuous situation. The effect of increased engine availability at this port is enhanced by the twisted pipeline behavior of carriers. Since carriers demand engines before they deploy instead of continuously as in the case of the continuous pipeline, the engines which are stocked at the port in anticipation of carrier demands may also be used for the port's flying activity. The results of this comparison appear in table 2.

Table 2 also shows that the CV-PAC-1 carrier performs significantly worse in the twisted pipeline. It was noted that this carrier supports a more extensive flying activity with a yearly failure rate of 28.17 engines than other system carriers. These other carriers produce 3 or 4 failures per year.

The statistical comparison between 1) and 3) compounds the situation of ports <u>and</u> twisted carrier pipelines. Once again, the port and its most active carrier demonstrate significantly different performances (table 3).

Another system, identified by engine model type J52P - series 8A and 8B in the Jacksonville Naval Aircraft Facility System (Airlaut Command, Combination Codes 102 and 104) was observed in the same manner. Stock levels close to DOD requirements were used for each base and in both the

continuous and the twisted pipeline situations were run (including ports). The three carriers in this system were particularly active, with 51.6 yearly engine failures. As table 4 shows, carriers are significantly different in terms of their performance measures in each configuration.

It should be noted that no other repair system locations exhibit criterion differences using this test of hypothesis for equal means in either of the two systems observed.

TABLE 1

Comparison of Mean Criterion--Continuous Pipeline Without Ports vs. Continuous Pipeline with Ports: Engine Model Type T58GE 8B, 8F

$\overline{x}_1$: mean criterion in Continuous Pipeline Configuration <u>without</u> non-flying/maintenance ports

$\overline{\sigma}_1$: standard deviation of $\overline{x}_1$

$\overline{x}_2$: mean criterion in Continuous Pipeline Configuration <u>with</u> non-flying/maintenance ports

$\overline{\sigma}_2$: standard deviation of $\overline{x}_2$

Base	Spare Stock Level	$\overline{x}_1$	$\overline{\sigma}_1$	$\overline{x}_2$	$\overline{\sigma}_2$	Test Statistic	Significant?
Adak	1	0.04428	0.0051	0.04197	0.00581	0.298	
Alameda	4	0.01397	0.00216	0.01179	0.00216	0.713	
Barbers Point	2	0.01145	0.00243	0.00685	0.00134	1.658	
Cubi Point	4	0.01314	0.00414	0.02821	0.0053	2.24	✓
CV-PAC-1*	3	0.01211	0.00236	0.01133	0.00331	0.1918	
CV-PAC-4*	1	0.006	0.00353	0.00315	0.00171	0.7266	
Imperial Beach	0	0.01163	0.00494	0.01559	0.0036	0.6478	
Kadena-AFB	0	0.03174	0.00924	0.0233	0.00703	0.7269	
Kauai-PMRF**	-	-	-	-	-	-	
NAV-PAC-1*	1	0.01931	0.00968	0.00587	0.0044	1.264	
NAV-PAC-2*	1	0.02074	0.01112	0.00652	0.00533	1.153	
NAV-PAC-3*	1	0.01609	0.00791	0.01117	0.00554	0.509	
NAV-PAC-4*	1	0.00815	0.00566	0.00650	0.00453	0.227	
NAV-PAC-5*	1	0.01894	0.009	0.01156	0.00546	0.701	
North Island	10	0.00473	0.00071	0.00494	0.00078	0.199	
Point Mugu	1	0.00328	0.00177	0.00289	0.00149	0.173	
Widfey Island	1	0.00771	0.00248	0.00436	0.00149	1.157	
Iwakuni-MC	1	0.01781	0.0051	0.02116	0.00447	0.4939	
Kaneohe-MC	1	0.00248	0.00105	0.00327	0.00115	0.507	

* Carrier

** Aircraft do not 'fly'

TABLE 2

Comparison of Mean Criterion--Continuous Pipeline With Ports vs. Twisted Pipeline With Ports: Engine Model Type T58GE 8B, 8F

$\bar{x}_1$: mean criterion in Continuous Pipeline Configuration with non-flying/maintenance ports

$\bar{\sigma}_1$: standard deviation of $\bar{x}_1$

$\bar{x}_2$: mean criterion in Twisted Pipeline Configuration with non-flying/maintenance ports

$\bar{\sigma}_2$: standard deviation of $\bar{x}_2$

Base	Spare Stock Level	$\bar{x}_1$	$\bar{x}_2$	$\bar{\sigma}_1$	$\bar{\sigma}_2$	Test Statistic	Significant?
Adak	1	0.04197	0.04707	0.00581	0.00672	0.5741	
Alameda	4	0.01129	0.01008	0.00216	0.00208	0.4035	
Barbers Point	2	0.00685	0.00770	0.00134	0.00165	0.39989	
Cubi Point	4	0.02821	0.00169	0.00530	0.00073	4.9569	✓
CV-PAC-1*	3	0.01133	0.03818	0.00331	0.00551	4.177	✓
CV-PAC-4*	1	0.00315	0.00847	0.00171	0.0034	1.3978	
Imperial Beach	0	0.01559	0.00890	0.0036	0.00353	1.326	
Kadena-AFB	0	0.02330	0.02111	0.00703	0.00655	0.2279	
Kauai-PMRF**	-	does not fly		-	-	-	
NAV-PAC-1*	1	0.00587	0.02187	0.0044	0.00856	1.662	
NAV-PAC-2*	1	0.00652	0.01802	0.00533	0.00594	1.441	
NAV-PAC-3*	1	0.01117	0.00460	0.00554	0.00342	1.009	
NAV-PAC-4*	1	0.0065	0.01122	0.00453	0.00551	0.661	
NAV-PAC-5*	1	0.01156	0.02503	0.00546	0.01214	1.012	
North Island	10	0.00494	0.00601	0.00078	0.00069	1.027	
Point Mugu	1	0.00289	0.00310	0.00149	0.00146	0.10067	
Widfey Island	1	0.00436	0.00435	0.00149	0.00118	0.00526	
Iwakuni-MC	1	0.02116	0.02605	0.00447	0.00575	0.6714	
Kaneohe-MC	1	0.00327	0.00703	0.00115	0.00209	1.576	

* Carrier
** Aircraft do not 'fly'

TABLE 3

Comparison of Mean Criterion--Continuous Pipeline Without Ports vs. Twisted Pipeline With Ports: Engine Model Type T58GE 8B, 8F

$\overline{x}_1$: mean criterion in Continuous Pipeline Configuration without non-flying/ maintenance ports

$\overline{\sigma}_1$: standard deviation of $\overline{x}_1$

$\overline{x}_2$: mean criterion in Twisted Pipeline Configuration with non-flying/ maintenance ports

$\overline{\sigma}_2$: standard deviation of $\overline{x}_2$

Base	Spare Stock Level	$\overline{x}_1$	$\overline{x}_2$	$\overline{\sigma}_1$	$\overline{\sigma}_2$	Test Statistic	Significant?
Adak	1	0.04428	0.04707	0.0051	0.00672	0.33	
Alameda	4	0.01397	0.01008	0.00216	0.00208	1.297	
Barbers Point	2	0.01145	0.00770	0.00243	0.00165	1.276	
Cubi Point	4	0.01314	0.00169	0.00414	0.00073	2.72	✓
CV-PAC-1*	3	0.01211	0.03818	0.00236	0.00551	4.349	✓
CV-PAC-4*	1	0.006	0.00847	0.00353	0.0034	0.503	
Imperial Beach	0	0.01163	0.00890	0.00494	0.00353	0.449	
Kadena-AFB	0	0.03174	0.02111	0.00924	0.00655	0.919	
Kauai-PMRF**	-	does not	fly	-	-	-	
NAV-PAC-1*	1	0.01931	0.02187	0.00968	0.00856	0.198	
NAV-PAC-2*	1	0.02074	0.01802	0.01112	0.00594	0.216	
NAV-PAC-3*	1	0.01609	0.0046	0.00791	0.00342	1.333	
NAV-PAC-4*	1	0.00815	0.01122	0.00566	0.00551	0.388	
NAV-PAC-5*	1	0.01894	0.02503	0.009	0.01214	0.4029	
North Island	10	0.00473	0.00601	0.00071	0.00069	1.293	
Point Mugu	1	0.00328	0.0031	0.00177	0.00146	0.078	
Widfey Island	1	0.00771	0.00435	0.00248	0.00118	1.22	
Iwakuni-MC	1	0.01781	0.02605	0.00510	0.00575	1.072	
Kaneohe-MC	1	0.00248	0.00703	0.00105	0.00209	1.945	

* Carrier

** Aircraft do not 'fly'

TABLE 4

Comparison of Mean Criterion--Continuous Pipeline With Ports vs. Twisted Pipeline With Ports: Engine Model Type J52P 8A, 8B

$\bar{x}_1$: mean criterion in Continuous Pipeline Configuration with non-flying/ maintenance ports

$\bar{\sigma}_1$: standard deviation of $\bar{x}_1$

$\bar{x}_2$: mean criterion in Twisted Pipeline Configuration with non-flying/ maintenance ports

$\bar{\sigma}_2$: standard deviation of $\bar{x}_2$

Base	Spare Stock Level	$\bar{x}_1$	$\bar{x}_2$	$\bar{\sigma}_1$	$\bar{\sigma}_2$	Test Statistic	Significant?
CV-LANT 1*	5	0.00357	0.01909	0.00109	0.00206	-6.6592	✓
CV-LANT 2*	5	0.00477	0.01535	0.00081	0.00159	-5.93	✓
CV-LANT 3*		0.00750	0.01947	0.00124	0.00204	-5.014	✓
Cherry Point	6	0.00897	0.00838	0.001	0.00078	0.4652	
South Weymouth	1	0.01809	0.01840	0.00153	0.00171	-0.135	
Oceana	25	0.02737	0.02624	0.00214	0.00178	0.4059	
Key West	1	0.00915	0.00893	0.00272	0.00267	0.0577	
Norfolk	2	0.01326	0.01360	0.00226	0.00232	-0.1049	
Paxtuxent Riv	1	0.00116	0.00323	0.0071	0.00183	-0.1895	
Willow Grove	1	0.02324	0.02118	0.00231	0.00204	0.668	
Memphis	1	0.0036	0.00644	0.00157	0.00241	-0.428	

* Carrier

VII

RECOMMENDATIONS AND GENERAL REMARKS

The previous section demonstrated the use of SPAERS in computing performance of repair system configurations.

Based on the runs of two actual repair system, it was observed that some carriers and ports in a twisted pipeline perform significantly differently from the same carriers and ports in the continuous case. The remainder of the system (in these cases) appeared to perform independently of carrier behavior.

It was also suggested that the extent to which a carrier in the twisted pipeline configuration will perform differently from the continuous one is somehow indicated by the carrier's failure rate. The nature of the twisted pipeline carrier activity resembles that of a periodic review inventory system where each inventory review occurs during a carrier's dock time. When this is contrasted with the continuous review analog for the continuous pipeline, inventory requirements for the twisted pipeline would be higher. If the carrier's flying activity (and so the failure rate) is low, the two inventory systems have similar stock requirements. On the other hand, if activity is high, the periodic review will require more stock to adequately support its flying.

Stronger statements about expected performances for a general system would require more extensive investigation.

The authors recommend the following:

1) A variety of systems should be compared in this way to support some general statements concerning carrier activity. DOD stock levels are adequate for this investigation.

2) NAVMET levels should be used as input stock levels to SPAERS simulation of repair systems, noting the difference between this system performance with DOD levels.

3) Ports and their activity with carrier operations should be considered in analytical spare stock level allocation to account for their role in carrier performance.

VIII APPENDICES

Appendix A: SPAERS Delineation

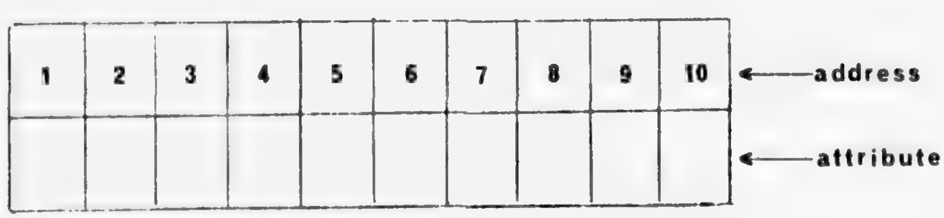
Contents

AI. Linked Listing/Queue Mechanism

Linked listing is a common technique used in SPAERS even scheduling. The invariant relation in a linked list is the fact that all the entries in the list are ordered according to some discipline. The method of ordering relies on a system of "pointers," or attributes of each list entry which indicates the address of the next entry in the list. Each entry is identified by only one address. Its position in the list is identified by the "pointers" or system links.

The application of linked lists to computer information management is crucial to efficiency. A new entry, for instance, does not require re-shifting any part of the list. Only two "pointers" need to be changed. Similarly, removal of an item from the list requires only "pointer" changes. The expected number of comparisons in a list search is the same as any other linear search (1/2 N, where N is the average size of the list).

Let the following illustration represent computer



storage in the form of an array.

For the purpose of this example, ordering attributes can be event times which will enter and leave the list.

To maintain the list ordering, an auxiliary array is required. Each entry in this array indicates the address of the entry following it in the list:

Schedule

address	0	1	2	3	4	5	6	7	←address
attribute (time)	9999	11:00	11:30	11:20	11:25				←attribute (time)

Link

address	0	1	2	3	4	5	6	7	←address
attribute	1	3	0	4	2	6	7	8	←attribute (address of the attribute immediately following)

EMPTY ADDRESS--5

active (addresses 0–4) empty (addresses 5–7)

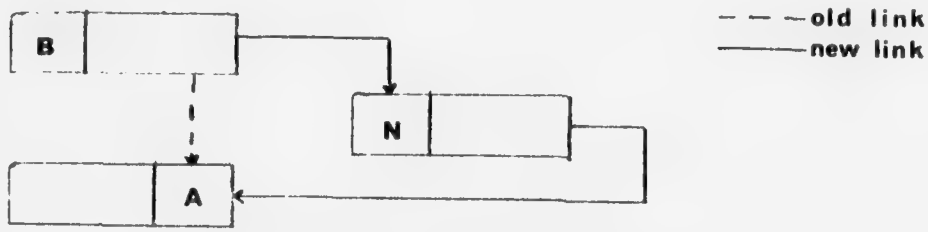
Note that the linked list may be divided into an active and an empty section. The active section consists of those addresses whose attributes are in the schedule. The empty section consists of those addresses whose attributes are not part of the active list. Both sections are linked and the leading element of both sections can be identified.

It is convenient to have the zero entry in the link system so that its link attribute identifies the address of

of the leading entry in the active list. The link of the final entry in the schedule is back to the zero address, thus forming a linked cycle. The zero address schedule attribute should be of a magnitude so that it will always be at the end of the schedule. In this way, the zero entry acts as a schedule "anchor" in that any other entry in the list must preceed it, and its link attribute always identifies the top of the list. (A special case worth mentioning occurs when the zero link attribute identifies its own address, signifying an empty schedule list.)

To add an element to a linked list:

1) enter the attribute in the first address available in the empty list, retaining the address.
2) change the empty address identifier to the next empty address.
3) identify the point of insertion for the new item by searching the ordering attribute list and utilizing the link system. The address of the preceeding entry will identify an insertion point (B in the figure).
4) link the new item with the old address link of B. Link B to the address of the new item. Insertion is complete and the invariant relation still holds.



To insert N between B and A

In deleting an element N from the list,

1) Locate the item to be deleted by identifying its address N and the address of its preceeding link B. (In event schedules, the outgoing item is usually the leading entry.)
2) Link B to A retaining the address of N.
3) Link N with the address of the first empty element and identify N as the new first empty element.

Variations on this technique involve the use of "backpointers" to facilitate a search which starts at the end of the list. Other attribute lists may be used to describe the elements which are found in a specific ordering.

SPAERS event scheduling relies on linked list systems (later referred to as queues). Engine availability, aircraft failure, and carrier events are each queued in their own link systems in order of their schedule occurrence. The leading entry in each link system is the next event which will occur of that type. In order to determine the next event, only the leading elements of each link system require comparison.

Aircraft failure queue

Any aircraft which is not in the down state will be assigned a failure date. All operational aircraft can be found in the failure queue and ordered by failure dates.

Most of the system aircraft are expected to be in an operational state at any point in time making the active portion of this queue large. It can be seen that by permanently addressing the aircraft's failure time by its actual aircraft number, maintenance of an empty queue can be avoided.

0	1	2	3	4	5	6	7	←a/c number
9999	20		23	26	34		9	←fail date

0	1	2	3	4	5	6	7	
7	3	4	4	5	0	7	1	←a/c with next fail date

Note that in the above queue link system, aircraft numbers "2" and "6" are in the down state and are not part of the link system.

Identifiers used in the failure queue.

ACFAILQFPT (N) is the address (a/c number) which will fail after aircraft N.

ACFAILQBPT (N) is the address of the aircraft which will fail before N.

ACFAILQBOT identifies the aircraft number to

fail last.

BASEQ identifies the base location of an a/c.

Availability Queue

Availability Queue sequences the times of engine availability throughout the system. Since engines are not specifically identified by number, they have no readily made address as in the case of the failure queue. An engine which enters the system pipeline as a result of a demand or a failure is queued in order of its availability date. The base at which an engine will become available is also an attribute associated with the address of a queued engine.

Relevant identifiers for this system include:

BASENGAVAIL (N)--the base where the engine identified by address N will become available.

ENGAVAILBPT (N)--the address of the engine which will become available in the system prior to the engine with address N.

ENGAVAILQFPT--the address of the engine which will become available in the system after the engine whose address is N.

ENGAVAILQNT--identifies the address of the first available array position.

ENGAVAILQHOLD--retains addresses during insertion cr extraction.

ENGAVAILQBOT--identifies the engine address

which will experience the last availability date.

ENGAVAILTIME (N)--availability date of the engine with address N.

Ship queue

Ship queue orders carrier events by date. All carriers will always occupy this queue, since they always have an event pending. In this case, the carrier number is its queue address.

Identifiers used in queue maintenance are:

SHIPTIME (N) is the time when the next event (dock or deploy) will occur for carrier N.

SHIPQPT (N) identifies the carrier which will experience an event after carrier N.

INSRT and LINSRT retain addresses during resinsertion process.

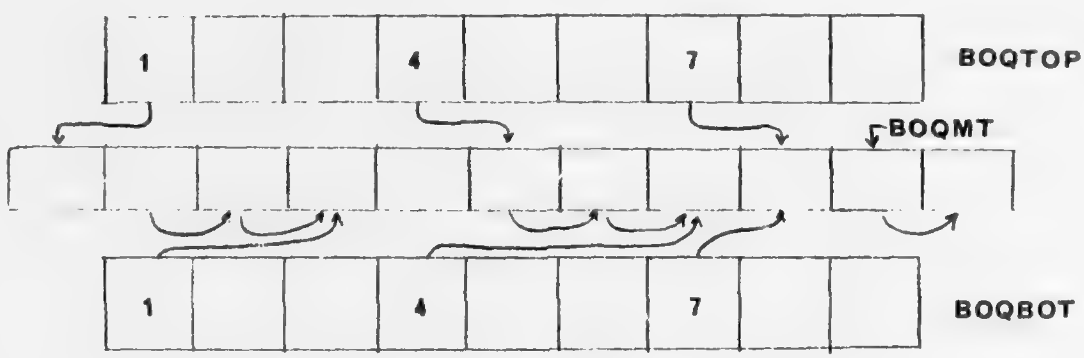
Note that SHIPQPT (O) identifies the carrier with the earliest future carrier event.

Backorders

It was stated earlier that demands that cannot be levelled (filled) immediately are logged as backorders to be filled on a first come first served basis as engine availability permits. This requires that the backorder commitments be queued at each base in the order of which they occur.

The queued attribute is that base to which the backorder is due.

In this case, queue insertion always occurs at the "bottom" of the list and removal is done at the "top." Thus only the leading and the final backordered base address need to be identified with the base which logged the backorder. The mechanism may be pictured as follows:



BOQTOP (N) identifies (points to) the address of Base N's first backorder commitment.

BOQBOT (N) identifies the address of the base N's last backorder commitment. (Note: If both the above variables are zero for some N, no backorders are outstanding at that base.)

A special case of the backorder situation occurs when a base must backorder to itself (its own flying activity). A self-backorder is the same as a downed aircraft, and is queued as both. Similar to the backorder case, down aircraft

are re-installed in the same order in which they occurred. The queuing mechanism has a similar operation to the back-order mechanism (see page A8). The difference with the down aircraft queue is that aircraft addresses are actual aircraft numbers (as in the case of the aircraft failure event schedule.

DACQPT (N) indicates the next aircraft in some base's down queue.

DACQTOP (N) identifies the first down aircraft number at base _N_ (zero if no down aircraft exist at base _N_).

DACQBOT (N) identifies the last down aircraft number at base _N_ (zero if no down aircraft exist at base _N_).

AII. SPAERS Flowchart

The task of developing a computer program is facilitated by preparing a layout of the logical flow of operations. This layout serves as a basis for which the source instructions are written.

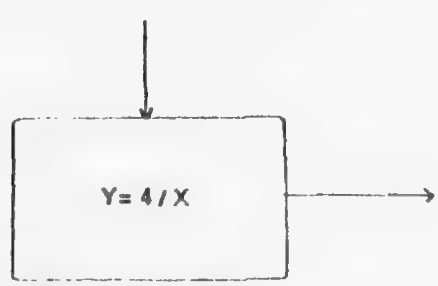
An assortment of symbols for the basic types of internal operations are used for the preparation of flowcharts.

SPAERS flowchart symbols are as follows:

1. The slot:

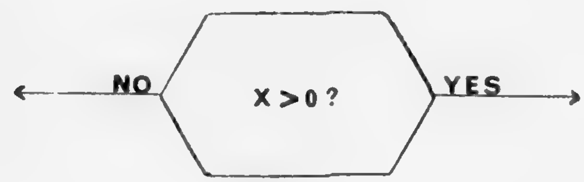
Used to denote the beginning or end of the program. It serves as a point of reference and does not represent any particular operation.

2. The rectangle:

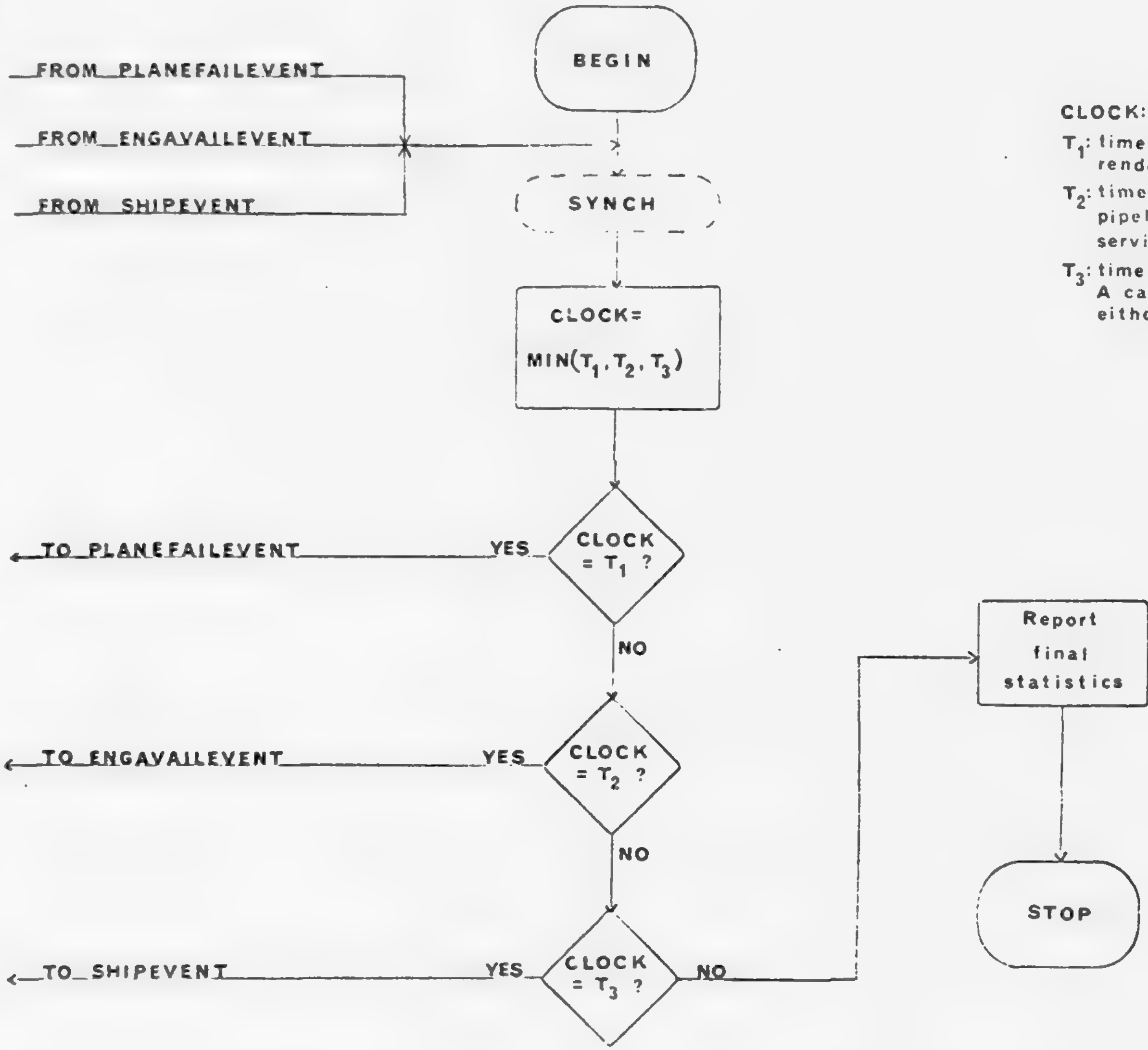


Used to indicate an internal operation which corresponds to one or more assignment statements. The contents of the box may be either of symbolic/ mathematical form or of narrative form.

3. A hexagonal shaped box:



Used to describe an internal operation requiring a decision. The decision point is stated as a yes-or-no question.



KEY

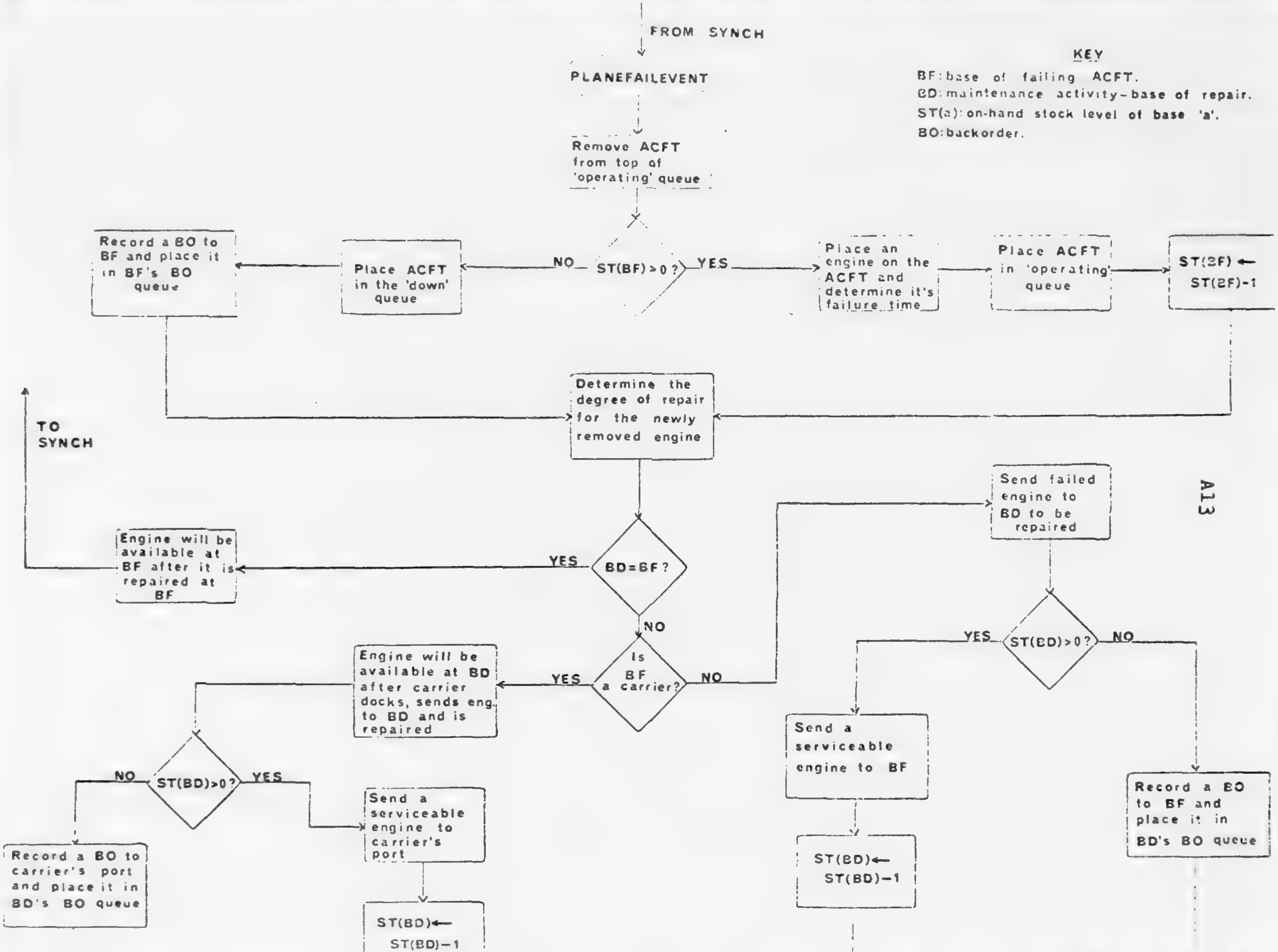
CLOCK: present time.

T_1: time at which an engine fails and renders an ACFT inoperative.

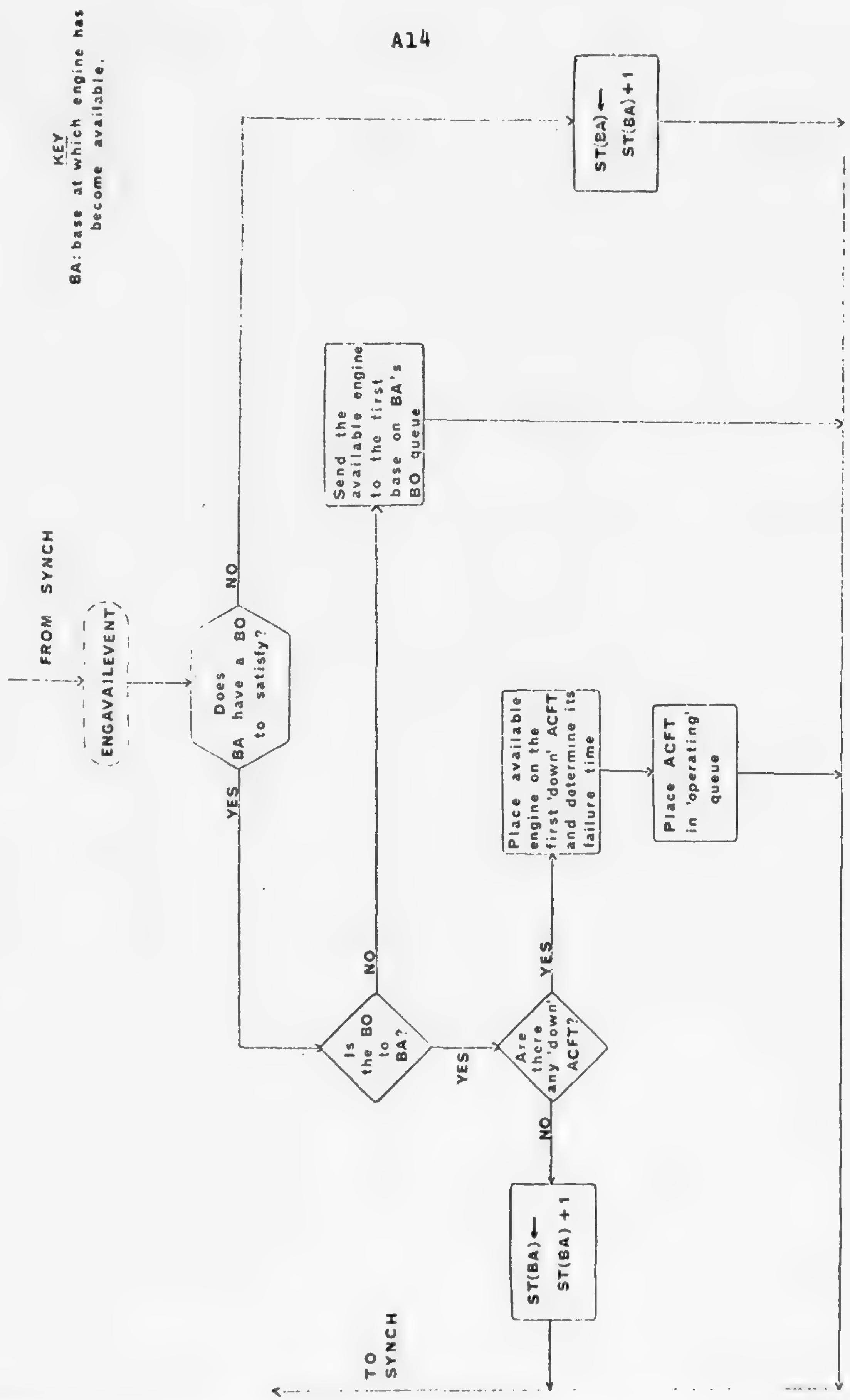
T_2: time at which an engine in a pipeline becomes available as serviceable on-hand stock.

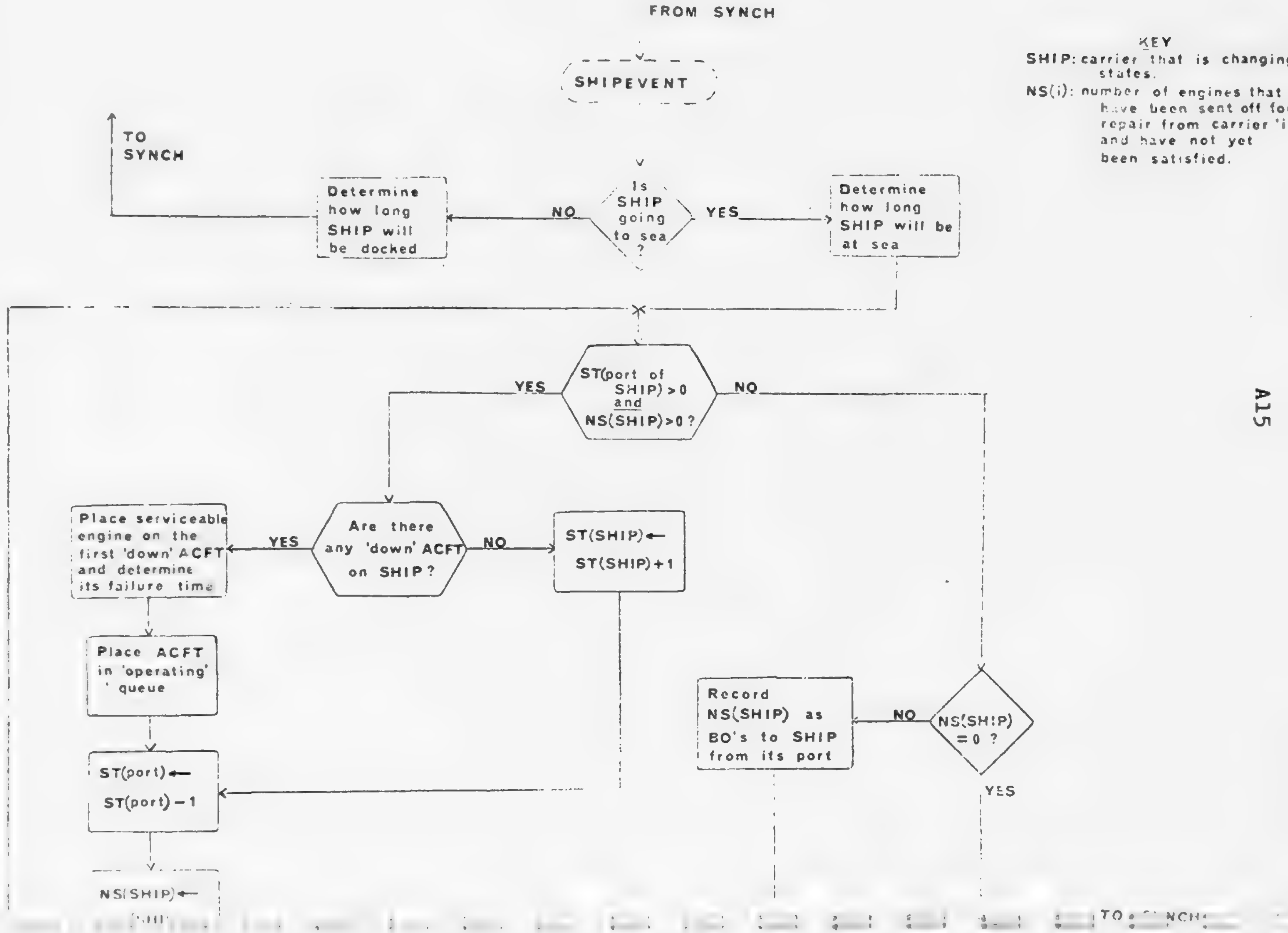
T_3: time of a carrier state change. A carrier state is defined as either 'AT SEA' or 'AT PORT'.

A13



FROM SYNCH
PLANEFAILEVENT
Remove ACFT from top of 'operating' queue
ST(BF)>0?
NO
YES
Place ACFT in the 'down' queue
Record a BO to BF and place it in BF's BO queue
Place an engine on the ACFT and determine it's failure time
Place ACFT in 'operating' queue
ST(BF) ← ST(BF)-1
Determine the degree of repair for the newly removed engine
TO SYNCH
BD=BF?
YES
NO
Engine will be available at BF after it is repaired at BF
Is BF a carrier?
YES
NO
Engine will be available at BD after carrier docks, sends eng. to BD and is repaired
ST(BD)>0?
NO
YES
Record a BO to carrier's port and place it in BD's BO queue
Send a serviceable engine to carrier's port
ST(BD)← ST(BD)−1
Send failed engine to BD to be repaired
ST(BD)>0?
YES
NO
Send a serviceable engine to BF
ST(BD)← ST(BD)−1
Record a BO to BF and place it in BD's BO queue
KEY
BF: base of failing ACFT.
BD: maintenance activity–base of repair.
ST(a): on-hand stock level of base 'a'.
BO: backorder.


KEY
BA: base at which engine has become available.
FROM SYNCH
ENGAVAILEVENT
Does BA have a BO to satisfy?
YES
NO
Is the BO to BA?
NO
YES
Send the available engine to the first base on BA's BO queue
Are there any 'down' ACFT?
YES
NO
Place available engine on the first 'down' ACFT and determine its failure time
Place ACFT in 'operating' queue
ST(BA) ← ST(BA) + 1
ST(BA) ← ST(BA) + 1
TO SYNCH


FROM SYNCH
SHIPEVENT
Is SHIP going to sea ?
YES
NO
Determine how long SHIP will be at sea
Determine how long SHIP will be docked
TO SYNCH
ST(port of SHIP) > 0 and NS(SHIP) > 0 ?
YES
NO
Are there any 'down' ACFT on SHIP ?
YES
NO
Place serviceable engine on the first 'down' ACFT and determine its failure time
ST(SHIP) ← ST(SHIP) + 1
Place ACFT in 'operating' queue
ST(port) ← ST(port) − 1
NS(SHIP) ←
NS(SHIP) = 0 ?
NO
YES
Record NS(SHIP) as BO's to SHIP from its port
KEY
SHIP: carrier that is changing states.
NS(i): number of engines that have been sent off for repair from carrier 'i' and have not yet been satisfied.

AIII. Description of Procedures

(i) PLANEFAILEVENT

This procedure is invoked at the time an engine fails which renders an aircraft inoperative.

<u>Nomenclature</u>:

- BASEFAILEDAC--the base number of the base at which aircraft has failed.
- DESTINBASE--the base number of the base at which failed engine is to be repaired.
- FAILEDAC--aircraft number of failed aircraft.
- FAILDEGREE--failure degree of engine
- HD(I)--the highest degree failure which base <u>I</u> can repair. ('4' denotes no maintenance activity at base <u>I</u>.)
- LIFENG(I,J)--the number of flying hours before failure of an engine in the $\underline{I}^{th}$ position on aircraft <u>J</u>. (I = 1, 2, · · · , N: N ≡ engine capacity.)
- NEXTECH(I,J)--base number of base that repairs base <u>I</u>'s degree <u>J</u> failed engines.
- REPAIRTIME(I)--repair time, in days, of an $\underline{I}^{th}$ degree failed engine.
- SHIPNUM(I)--the carrier (ship) number of base <u>I</u> ('0' if a ground base).

- STOCKLEVEL(I)--number of serviceable engines on hand at base I.
- CCDIST(I,J)--cumulative distribution of failure degree of combination code I, i.e., Pr {FAILDEGREE ≤ J} for J = -1, 0, 1, 2, 3.
- DUMMYTIME--time remaining in carrier's deployment if BASEFAILEDAC is a carrier in 'Twisted Pipeline Configuration.'

  --0 otherwise.

Synopsis:

--Determine FAILEDAC and BASEFAILEDAC from FAILTIME and BASEQ queues, respectively.

--Remove FAILEDAC from FAILTIME queue by appropriately rearranging the pointer mechanism.

--Determine FAILDEGREE:

- Generate a random variable, U, uniformly distributed between (0, 1).
- FAILDEGREE will then equal the largest integer K, such that {K: U ≤ CCDIST(I,K), K = -1, 0, 1, 2, 3}. I ≡ combination code of the aircraft type of FAILEDAC.

--Attempt to make FAILEDAC operational.

- If STOCKLEVEL (BASEFAILEDAC) > 0, then:
  1) Remove a repaired engine from stock and place the engine on FAILEDAC.
  2) Determine the life of the engine, LIFENG

(1, FAILEDAC) (i.e., number of flying hours before failure), by invoking the RNG function procedure (see (x) below).

3) Determine the next failure time (in days) of the aircraft by invoking FAILGEN procedure with parameters: (FAILEDAC, SHIPNUM(BASEFAILEDAC), BASEFAILEDAC) (see (vii) below).

- If STOCKLEVEL(BASEFAILEDAC) = 0, then:

  1) Place FAILEDAC in the down aircraft queue utilizing DACQPT, DACBOT and DACTOP.

  2) Record a backorder to BASEFAILEDAC from BASEFAILEDAC by utilizing BASEBO queue.

  3) Queue backorder accordingly by invoking BACKORDQ procedure with parameter: (BASEFAILEDAC) (see (ix) below).

--Engine Transaction

- If HD(DESTINBASE) > FAILDEGREE, i.e., DESTINBASE cannot repair the engine as in the case of a port which has neither maintenance nor flying activity but is required to serve as an intermediary point for all its carriers (user option--see Appendix B), then provisions must be made for demand accounting:

  1) The transaction of the failed engine will be between the port and the

appropriate maintenance base.

2) BASEFAILEDAC will demand an engine from the port immediately in the "Continuous Pipeline" configuration or at the time of its next deployment in the "Twisted Pipeline" configuration.

- If BASEFAILEDAC cannot repair the failed engine, i.e., BASEFAILEDAC $\neq$ DESTINBASE, then:

1) Schedule the availability of the engine at DESTINBASE by invoking AVAILQ procedure (see (vi) below) with parameters: (DESTINBASE, BASEFAILEDAC, SHIPNUM(BASEFAILEDAC), REPAIRTIME(FAIL-DEGREE)).

(Note: In the case of a port which serves as an intermediary point for its carriers, provisions are made to include the time remaining in a carrier's deployment by using DUMMYTIME.)

2) If STOCKLEVEL(DESTINBASE) > 0, then schedule the availability of a serviceable engine of DESTINBASE stock at BASEFAILEDAC by invoking AVAILQ procedure (see (vi) below) with parameters: (BASEFAILEDAC, DESTINBASE, 0,0).

If on-hand stock does not exist at

DESTINBASE, then record a backorder to BASEFAILEDAC from DESTINBASE by utilizing BASEBO queue. Queue backorder accordingly by invoking BACKORDQ procedure with parameter: (DESTINBASE) (see (ix) below).

- If BASEFAILEDAC can repair the failed engine, i.e., BASEFAILEDAC = DESTINBASE, then: schedule the availability of this engine at BASEFAILEDAC by invoking AVAILQ procedure with parameters: (BASEFAILEDAC, BASEFAILEDAC, 0, REPAIRTIME (FAILDEGREE)).

(ii) ENGAVAILEVENT

This procedure is invoked at the time an engine in a particular pipeline becomes available as on-hand serviceable stock.

Nomenclature:

- BASEAVAIL--the base number of the base at which an engine has become available.

Synopsis:

--Determine BASEAVAIL from BASENGAVAIL queue and rearrange pointer mechanism.

--If BASEAVAIL has a backorder to satisfy, then: (The following is performed utilizing backorder queue mechanisms.)

- If the backorder exists to BASEAVAIL, then: If there are any down aircraft, then:

  1) Place engine on first aircraft in down

aircraft queue.

2) Determine the life of the engine (flying hours).

3) Determine the next failure time of the aircraft.

If there are no down aircraft at BASEAVAIL, then place the available engine in BASEAVAIL's on-hand stock.

- If the backorder exists to an outside base, then schedule the availability of the engine at the backordered base by invoking AVAILQ procedure (see (vi) below) with parameters: (backordered base, BASEAVAIL, 0, 0).

--If there are no backorders to satisfy, place available engine in BASEAVAIL's on-hand stock.

(iii) SHIPEVENT

This procedure is invoked at the time the state of a particular carrier changes.

Nomenclature:

- BASENUM (I)--base number of carrier (ship) number I.
- PORTSCHED (I,J), SEASCHED (I,J)--dock time, deployment time of carrier (ship) number I during J^{th} schedule: J = 1, 2, • • • , schedule length. (Note: schedule may be either deterministic or probabilistic--user option: see Appendix B.)
- SHIPTIME (I)--time of next state change of carrier

(ship) number I.

• STATE (I)--current state of carrier number I.

Synopsis:

--Determine which carrier is changing states by utilizing SHIPQPT mechanism.

--If this carrier's state change is from 'AT SEA' to 'AT PORT,' then:

- Determine how long it will be docked and set SHIPTIME accordingly.

--If the carrier's state change is from 'AT PORT' to 'AT SEA,' then:

- Determine how long it will be at sea and set SHIPTIME accordingly.
- Carrier's port attempts to restock carrier with the number of engines sent out during last deployment plus any backorders from any previous deployments.

  The order of restocking is as follows:

  1) Rejuvenate any down aircraft on board carrier and determine the life of the new engine plus the failure time of the aircraft.
  2) If there are no down aircraft, then place engine in carrier's on-hand stock.

  If at any time during the restocking the port's stock is depleted, the appropriate number of

backorders are recorded at the port.

--SHIPTIME (carrier) is appropriately queue by rearranging SHIPQPT mechanism accordingly.

(iv) SAMPLE

SAMPLE procedure is invoked every SI simulation days where SI is the proposed current independent time length (see Appendix B).

Nomenclature:

- NUMSAMPS--identifies the current sample number.
- TOTSAMPLES--total number of samples desired (user option--see Appendix B).
- PASSFAIL (K)--is 1 if samples taken at base K passed the independence test; is 0 otherwise.
- PASSBASE--the number of bases which passed.
- TOTNUMACB--the number of bases which support their own flying activity.
- PF--the proportion of aircraft flying bases which passed the independence test.
- SAMPSEATIME (I,J)--the amount of time which carrier number I has spent in a deployed state during sample number J.

Synopsis:

Its main function is to segregate average down aircraft observations into independent observations. If the specified number of samples were taken, SAMPLE arranges for their independence tests.

When SAMPLE is called all time distributions and time related measurements are updated to this point. The sample number is then advanced.

If SAMPLE is called to terminate the final observation (NUMSAMPS = TOTSAMPLES), each base's down aircraft observations are tested for independence. The results of the test are found in the PASSFALL array.

The proportion of passing bases is calculated and then program control is sent to IND_OUT (see (xiv)). If control returns to SAMPLE, more observations are required and further sampling is executed.

(v) REPORT

This procedure is invoked at the termination of the simulation run. Final statistics are reported to the user.

(vi) AVAILQ

Parameters: (BASETO, BASEFROM, SHIPINDEX, REPTIME)
This procedure is invoked from either PLANEFAILEVENT or ENGAVAILEVENT.

Synopsis:

AVAILQ queues the time at which an engine becomes available at BASETO utilizing the ENGAVAILTIME and BASENGAVAIL queue mechanisms.

The availability time may be of the form:

- Time of engine availability =
  Present time plus travel time from BASEFROM to

BASETO plus REPTIME. REPTIME can take on any positive value. For example, in the case of transporting a new engine from one base to another, REPTIME = 0. In the case of transporting a failed engine to be repaired, REPTIME = repair time of failed engine.

- SHIPINDEX > 0 designates that BASEFROM is a carrier and is sending a failed engine to be repaired at BASETO. Since a carrier plane failure occurs only during deployment, an engine cannot be transported to BASETO until the carrier has docked. Thus, time of engine availability = SHIPTIME (carrier) plus travel time from the port of the carrier to BASETO plus repair time of the failed engine.

(vii) FAILGEN

Parameters: (PLANEN, SHIPN, BASEN)

This procedure is invoked from PLANEFAILEVENT, ENGAVAILEVENT or SHIPEVENT.

Nomenclature:

- DAYSTILFAIL--number of days before a failed engine causes PLANEN to become inoperative.
- FAILTIME--failure time of PLANEN.
- FLYHRS--flying hours of the engine with the minimum life on PLANEN.
- FOHM (I,J)--flying hours per month of aircraft type I at base number J.

- LIFENG (I,J)--number of flying hours before failure of an engine in the $\underline{I}^{th}$ position on aircraft $\underline{J}$. (I = 1, 2, • • • , N: N ≡ engine capacity.)

Synopsis:

FAILGEN is called at the time an aircraft is removed from the 'down aircraft' queue and made operational.

--Determine the position $\underline{i}$ of the engine with the minimum life, i.e., FLYHRS = LIFENG (i, PLANEN) and place this engine in the first position.

--Decrease the life of all engines on PLANEN by FLYHRS.

--Determine DAYSTILFAIL from the expression FLYHRS/ FOHM (PLANEN type, BASEN).

--If SHIPN > 0 (BASEN is a carrier), then we have to 'pad' DAYSTILFAIL so as to make sure PLANEN fails during deployment.

--Set FAILTIME = DAYSTILFAIL plus present time and queue this time accordingly utilizing ACFAIL queue mechanism.

(viii) DISTRIBUTIONS

Parameters: (BASE, DIST)

This procedure is invoked from either PLANEFAILEVENT, ENGAVAILEVENT, SHIPEVENT or SAMPLE.

Synopsis:

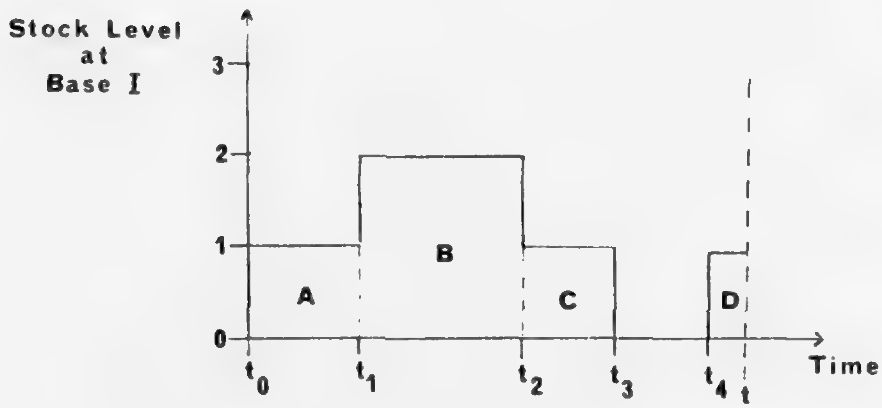
DISTRIBUTIONS is called at the time the present stock level, number of backorders or number of down aircraft

is about to change at any base.

DISTRIBUTIONS updates the weighted time frequency distribution and the area under frequency distribution curve at BASE.

Measurements:

1) STOCKDIST (I,J) = the amount of time spent at stock level J at base I.



at time t, STOCKDIST (I,1) = $(t_1 - t_0) + (t_3 - t_2) + (t - t_4)$.

2) STOCKAREA (I) = area under the frequency distribution curve of base I.

See above figure:

at time t, STOCKAREA (I)

$$= (t_1-t_o)\cdot 1 + (t_2-t_1)\cdot 2 + (t_3-t_2)\cdot 1 + (t_4-t_3)\cdot 0 + (t-t_4)\cdot 1$$

$$= A + B + C + D$$

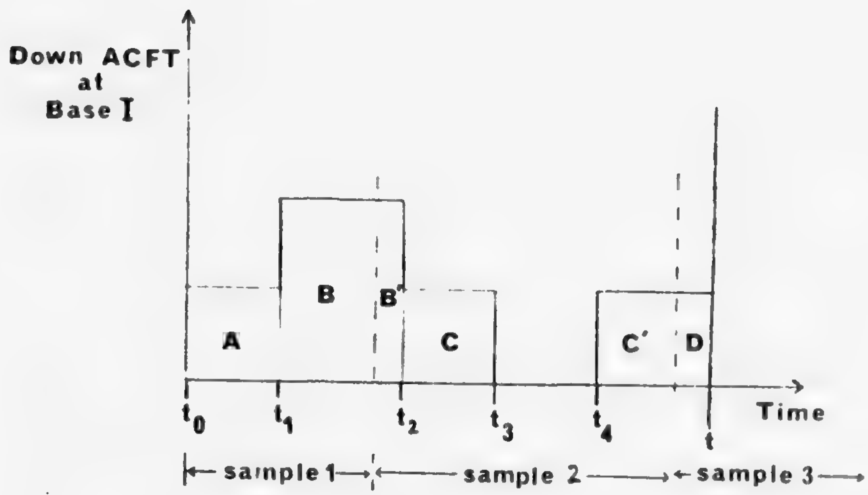
3) BODIST (I,J) = the amount of time spent at backorder level _J_ at a base _I_.

4) BOAREA (I) = area under the frequency distribution curve of base _I_.

5) DACDIST (I,J) = the amount of time spent at down aircraft level _J_ at base _I_ which has flying activity.

6) DACAREA (I,K) = area under the frequency distribution curve of base _I_ which supports flying activity during sample _K_.

By dividing the total simulation time in equal samples and testing for independence among the samples, we hope to perform statistical analysis on the 'criterion value' at each base which is obtained directly from DACAREA. The test for independence among samples is crucial for obtaining variances and confidence intervals (see Appendix D for a discussion on the independence test). (Note: criterion value ≡ average down aircraft at base _i_ / total number of aircraft at base _i_.)



at time $\underline{t}$, DACAREA (I,1) = A + B

DACAREA (I,2) = B' + C + C'

DACAREA (I,3) = D

DIST is the parameter that indicates which calculations are to be performed. The values it may take on are as follows:

- 'STOCK'--indicates stock level measurements are to be updated.
- 'BO'--indicates backorder measurements are to be updated.
- 'DAC'--indicates down aircraft measurements are to be updated.

(ix) BACKORDQ

Parameter: (BASE)

This procedure is invoked from PLANEFAILEVENT when a backorder is recorded at BASE.

BACKORDQ appropriately queues the backorder.

(x) RNG

Parameters: (MEAN, U)

This function procedure is invoked from either PLANEFAILEVENT, ENGAVAILEVENT or SHIPEVENT.

Synopsis:

RNG returns an exponentially distributed random variable with mean MEAN.

This is achieved by using the "inverse transformation" technique of random number generation:

- Generate a random number, U, uniformly distributed between (0,1).
- Return the value of F'(U) where F'(U) ≡ inverse of the exponential distribution function.

RNG is used to generate the number of flying hours until failure of an engine on an aircraft. MTBR of the combination code of the aircraft, where MTBR ≡ mean time between removal, is the argument which corresponds to MEAN.

(xi) SHIPTIME_RNG

Parameters: (MU, SIGMA, U)

This function procedure is invoked from SHIPEVENT.

Synopsis:

SHIPTIME_RNG returns a normally distributed random variable with mean MU and standard deviation SIGMA.

This is achieved by using one of the most

efficient stochastic variate generation techniques for a normally distributed random variable. This technique, like most random number generation techniques, is based on the generation of a random number, U, uniformly distributed between (0,1).

SHIPTIME_RNG is used when probabilistic 'dock' and 'deployment' carrier times are desired (user option--see Appendix B).

The arguments corresponding to MU and SIGMA for each carrier are user inputs.

(xii) SAFETY_STOCK

Parameters: (LAMDA, C)

This function procedure is invoked once for each base during the initialization stage.

Synopsis:

SAFETY_STOCK returns the largest integer, S, such that:

$$\text{Probability } \{x \leq S\} \geq C,$$

where x is defined as the number of engines in resupply. We assume that the probability distribution of x is Poisson with mean LAMDA. Thus, C is the desired probability of having no outstanding backorders at an arbitrary point in time.

- LAMDA = Daily Demand Rate x Average Resupply Time
- C is a user input for each base.

SAFETY_STOCK is used in the event a 'DOD-requirement' calculated spare stock level is desired instead of a user inputed level (user option--see Appendix B).

(xiii) IND_TEST

Parameters: (DACAREA, CRIT_BASE)

IND_TEST is a function procedure invoked from SAMPLE.

Nomenclature:

- XBAR--the average of all sample values taken at the base.
- DACAREA--a one dimensional vector in this procedure consisting of ordered sample observations.
- CRIT_BASE--the base whose observations are being tested.
- SUM 1--the statistic $\sum(x_i - x_{i+1})^2$
- SUM 2--the statistic $\sum(x_i - \overline{x})^2$
- CN(K)--the statistic $1 - \frac{\text{SUM1}}{2 \cdot \text{SUM2}}$ at base K.

Synopsis:

IND_TEST tests the observations at CRIT_BASE for independence and returns a value of "1" if the series is independent and "0" otherwise.

IND_TEST uses Fishman's technique of time series independence testing [3] (see Appendix D for discussion).

(xiv) IND_OUT

IND_OUT is invoked from SAMPLE, and determines

whether enough bases have passed independence tests (user option)

Nomenclature:

- NUMFAILS--the number of times the independence test has failed.
- LIMTEST--the user's limit on number of tests desired.

Synopsis:

If user specifications were met, test results are printed, listing number of samples, sample interval, bases, their pass/fail status, and its test statistic calculated by IND_TEST. The report of entire simulation statistics then follows.

Otherwise, if more tests are allowed, successive observations of downed aircraft are paired and regrouped, sample intervals are doubled, and TOTSAMPLES/2 more samples are regrouped. Again test results are printed as in the above paragraph.

If the limit on tests was reached, test results are printed in addition to some warnings and suggestions for achieving acceptance of independence. A report is then printed.

(xv) RES_OUT

RES_OUT is invoked from IND_OUT. It simply prints the results of the independence test each time the test is performed. It includes the base name, a pass/fail status, and the CN test statistic for each base.

AIV. SPAERS Source Instructions

```
*PL/C ID='GIVRAY,HENRY',TIME=(8,00),PAGES=100,LINES=5000

*OPTIONS IN EFFECT*     TIME=(8,0),PAGES=100,LINES=5000,NOATR,NOXREF,FLAGW,BNDRY,NOCMNTS,SORMGIN=(2,72,1),
*OPTIONS IN EFFECT*     ERRORS=(50,50),TABSIZE=29412,UDEF,SOURCE,OPLIST,NOCMPRS,HDRPG,AUXIO=10000,LINECT=60,NOALIST,
*OPTIONS IN EFFECT*     MCALL,MTEXT,DUMP=(S,F,L,E,U,R),DUMPE=(S,F,L,E,U,R),DUMPT=(S,F,L,E,U,R)

 NAVY: PROCEDURE OPTIONS(MAIN);                                               PL/C-R7.1--66 05/20/75 13:21 PAGE   1

 STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

   1                          NAVY: PROCEDURE OPTIONS(MAIN);

                                /*************************************************************/
                                /*                     S      SIMULATION    S                 */
                                /*                R            FOR THE          P             */
                                /*            E               PERFORMANCE            A        */
                                /*        A                      OF                      E    */
                                /*     P                   AIRCRAFT ENGINE                 R  */
                                /* S                        REPAIR SYSTEMS                   S */
                                /*THIS SIMULATION MODEL WAS DESIGNED TO BE A TOOL FOR MEASUR-  */
                                /*ING THE PERFORMANCE OF AIRCRAFT ENGINE REPAIR SYSTEMS OF THE */
                                /*UNITED STATES NAVY AS A FUNCTION OF ITS INITIAL STOCK        */
                                /*STRATEGIES.  THE MODEL DESIGN WAS PROMPTED BY THE NAVMET     */
                                /*PUBLICATION WRITTEN BY DR. JOHN A. MUCKSTADT WHICH ANALY-    */
                                /*TICALLY DETERMINES STOCK REQUIREMENTS AT EACH BASE IN A REPAIR*/
                                /*SYSTEM.  THE SIMULATION HAS THE ABILITY TO EVALUATE SYSTEM   */
                                /*PERFORMANCE OF ANY DISTRIBUTION OF SPARE ENGINE STOCK        */
                                /*THROUGHOUT THE SYSTEM, BOTH IN A TWISTED PIPELINE CONFIGUR-  */
                                /*ATION AND A STEADY STATE RENDERING OF A REPAIR SYSTEM.  DE-  */
                                /*FAULT INITIALIZATION OF SPARES FOLLOWS THE DOD REQUIREMENTS  */
                                /*FOR PIPELINE AND SAFETY SPARES. SYSTEM PERFORMANCE, IN TERMS */
                                /*OF TIME WEIGHTED AVERAGES OF DOWNED AIRCRAFT DIVIDED BY TOTAL */
                                /*SYSTEM AIRCRAFT, IS SUPPORTED BY OTHER OBSERVED STATISTICS   */
                                /*LIKE AVERAGE ON HAND STOCK, BACKORDERS AND DOWNED AIRCRAFT,  */
                                /*AND INCLUDES TIME WEIGHTED LISTINGS OF STOCK AND BACKORDER   */
                                /*LEVELS.                                                      */
                                /*   MODEL ASSUMPTIONS AND DESCRIPTIONS ARE AVAILABLE IN THE   */
                                /*MASTERS OF ENGINEERING PROJECT THESIS BY HENRY S. GIVRAY AND */
                                /*ROBERT A. SLON, CORNELL UNIVERSITY OR/IE DEPARTMENT, 1976.   */
                                /*VALUABLE CONSULTATION WAS PROVIDED BY DR. JOHN A. MUCKSTADT, */
                                /*CORNELL UNIVERSITY AND DR. JAMES MATTHESEN, U.S. NAVY SYSTEMS */
                                /*ANAYLYST.                                                    */
                                /*************************************************************/

   2    1          1            DCL(TOTNUMBASES,TOTNUMSHIPS,TOTNUMAC,TOTNUMTYPEAC,TOTNUMCC,
                                    MAXENGCAP,III,II2,MAXSCHEDLENGTH,TOTSAMPLES) BIN FLOAT;
   3    1          1            DCL(ENGMODEL,DEPOT) CHAR(20) VAR;
   4    1          1            DCL RNG ENTRY(BIN FLOAT,BIN FLOAT) RETURNS(BIN FLOAT);
   5    1          1            DCL SHIPTIME_RNG ENTRY(BIN FLOAT,BIN FLOAT,BIN FLOAT)
                                RETURNS(BIN FLOAT);
   6    1          1            DCL SAFETY_STOCK ENTRY(BIN FLOAT,BIN FLOAT) RETURNS(BIN FLOAT);
   7    1          1            GET LIST(ENGMODEL,DEPOT,TOTNUMBASES,TOTNUMSHIPS,
                                    TOTNUMAC,TOTNUMTYPEAC,TOTNUMCC,MAXENGCAP,
```

AD-A035 737 CORNELL UNIV ITHACA N Y DEPT OF OPERATIONS RESEARCH F/G 21/5
SPAERS: SIMULATION FOR THE PERFORMANCE AIRCRAFT ENGINE REPAIR S--ETC(U)
MAY 76 H GIVRAY, R SLON N00014-75-C-1172

UNCLASSIFIED NL

2 OF 3
AD A
035 737

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                    MAXSCHEDLENGTH,TOTSAMPLES);
   8    1          1             IF TOTNUMSHIPS=0 THEN
   9    1          1                DO;
  10    1    1     1                   III=0;
  11    1    1     1                   II2=0;
  12    1    1     1                END;
  13    1          1             ELSE
                                    DO;
  14    1    1     1                   III=1;
  15    1    1     1                   II2=MAXSCHEDLENGTH;
  16    1    1     1                END;
  17    1          1          B1:BEGIN;
                                 /*CURRENT LEVEL OF AVAILABLE STOCK AT EACH BASE                 */
  18    2          2             DECLARE(STOCKLEVEL(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*NUMBER OF DEMANDS PLACED ON A BASE'S SUPPLY ACTIVITY.          */
                                 DEMANDS(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*NUMBER OF DEMANDS WHICH ARE IMMEDIATELY SATISFIED.            */
                                 DEMSAT(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*TOTAL NUMBER OF BACKORDERED DEMANDS AT A BASE                 */
                                 BO(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*TOTAL NUMBER OF DOWNED A/C WHICH OCCUR AT A BASE.             */
                                 DAC(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*A SPECIFIED WEIGHT APPLIED TO PERFORMANCE MEASURES AT BASE I  */
                                 WEIGHT(TOTNUMBASES),

                                 /*THE CRITERION REQUIRED AT A BASE ACCORDING TO NAVY SPECIFICA- */
                                 /* TION.                                                        */
                                 DES_CRIT(TOTNUMBASES),

                                 /*THE TIME WEIGHTED AVERAGE NUMBER OF DOWNED A/C PER TOTAL      */
                                 /*NUMBER OF A/C.                                                */
                                 ACT_CRIT(TOTNUMBASES),

                                 /*DESIGNATES THE INITIALIZATION OF PIPELINE STOCK               */
                                 PLIND(TOTNUMBASES),

                                 /*THE AMOUNT OF STOCK IN A BASE'S PIPELINE                      */
                                 PLSTOCK(TOTNUMBASES),
                                 MAXPL(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*HIGHEST DEGREE OF REPAIR PERFORMED AT A BASE.                 */
                                 HD(TOTNUMBASES) INIT((TOTNUMBASES)0),

                                 /*CALCULATED APPROXIMATE NUMBER OF SPARES IN THE SYSTEM: FOR    */
                                 /* DECLARATION PURPOSES.                                        */
                                 SPARES INIT(0),

                                 /*STANDARD DEVIATION (IN DAYS) OF A SHIP'S MEANPORT AND MEAN    */
                                 /*SEA TIME.                                                     */
                                 STD_DEV(III:TOTNUMSHIPS),
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                 /*INDICATES A SPECIFIED PROBABILITY OF MEETING DEMAND              */
                 CRITICAL(TOTNUMBASES),

                 /*TOTAL TIME A SHIP HAS SPENT AT SEA.                               */
                 /*LAST SYSTEM TIME AT WHICH A SHIP LEFT PORT.                        */
                 (CUMSEATIME(III:TOTNUMSHIPS),LSEATIME(III:TOTNUMSHIPS))
                 INIT((TOTNUMSHIPS)0),

                 /*SUMMATION OF THE PRODUCTS X*TIME IN X,                            */
                 /*FOR BACKORDERS AND STOCKLEVELS .                                  */
                 (STOCKAREA(TOTNUMBASES),BOAREA(TOTNUMBASES)) INIT((TOTNUMBASES)0),

                 /*TIME WEIGHTED AVERAGE OF DOWNED A/C                               */
                 AVGDAC(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 /*LAST TIME THE STOCKLEVEL CHANGED AT A BASE                        */
                 /*LAST TIME THE DOWNED A/C COUNT CHANGED AT A BASE                  */
                 /*LAST TIME A BACKORDER LEVEL CHANGED AT A BASE                     */
                 LSTIME(TOTNUMBASES) INIT((TOTNUMBASES)0),
                 LDACTIME(TOTNUMBASES) INIT((TOTNUMBASES)0),
                 LBOTIME(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 /*CURRENT NUMBER OF DOWNED A/C AT A BASE                            */
                 DACNUM(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 /*TOTAL NUMBER OF A/C AT A BASE                                     */
                 AIRCRAFT(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 PASSBASE INIT(0),
                 /* INDEX OF INDEPENDENCE RESULT FOR EACH BASE                       */
                 PASSFAIL(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 PERCPASS,TOTNUMACB INIT(0),
                 /*TOTAL NUMBER OF ENGINES AT A BASE.                                */
                 ENGINES(TOTNUMBASES) INIT((TOTNUMBASES)0),                          .


                 /* DOWN AIRCRAFT SAMPLE OBSERVATIONS                                */
                 DACAREA(TOTNUMBASES,TOTSAMPLES) INIT ((TOTSAMPLES*TOTNUMBASES)0),

                 /*INDEPENDENCE TEST STATISTIC AT EACH BASE                          */
                 CN(TOTNUMBASES) INIT((TOTNUMBASES)0),

                 /* NUMBER OF FAILURES IN INDEPENDENCE TEST                          */
                 NUMFAILS INIT(0),

                 /* T STATISTIC FOR CONFIDENCE INTERVALS                             */
                 TSTAT,

                 /* LIMIT ON NUMBER OF INDEPENDENCE TESTS TO BE PERFORMED            */
                 LIMTEST,

                 /* DESIRED LEVEL FOR INDEPENDENCE TEST                              */
                 NORMAL_STAT,
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

        /* CUMULATIVE SEATIME IN EACH SAMPLE                          */
        SAMPSEATIME(1:1:TOTNUMSHIPS,TOTSAMPLES)
           INIT((TOTNUMSHIPS*TOTSAMPLES)0),

        /* BASE WHERE DOWN AIRCRAFT OBSERVATIONS WILL BE TESTED FOR IND.*/
        CRIT_BASE,

        /* CONFIDENCE LEVELS FOR INTERVALS AND INDEPENDENCE TEST     */
        ALPHAT, ALPHANORM,

        /* VARIANCE OF CRITERION AT EACH BASE                         */
        VARCRITERION(TOTNUMBASES) INIT((TOTNUMBASES)0),

        /*COMBINATION CODE (SEE MANUAL)                               */
        CC_CODE(TOTNUMCC),

        /*THE INPUT RATE IN ENGINES PER DAY CONTRIBUTED BY BASE I TO  */
        /*REPAIR ACTIVITY J.                                          */
        P(TOTNUMBASES,-1:3) INIT((TOTNUMBASES*5)0),

        /*STOCK IN PIPELINE                                           */
        R(TOTNUMBASES) INIT((TOTNUMBASES)0),

        /*CUMULATIVE DISTRIBUTION FOR P(FAILURE IS JTH DEGREE)        */
        /*FOR EACH COMBINATION CODE.                                  */
        CCDIST(TOTNUMCC,-1:3),

        /*DENSITY DISTRIBUTION FOR P(FAILURE IS JTH DEGREE).          */
        CCDISTDENS(TOTNUMCC,-1:3) INIT((TOTNUMCC*5)0),

        /*DESTINBATION BASE OF THE ENGINE WHICH HAS JUST FAILED.      */
        DESTINBASE,

        /*DESTINATION OF FAILURE J AT BASE I.                         */
        NEXTECH(TOTNUMBASES,-1:3),

        /*QUEUE MECHANISMS FOR BACKORDERS, DOWNED A/C.                */
        DACQBOT(TOTNUMBASES) INIT((TOTNUMBASES)0),
        BOQBOT(TOTNUMBASES) INIT((TOTNUMBASES) 0),
        BOQTOP(TOTNUMBASES) INIT((TOTNUMBASES)0),
        DACQTOP(TOTNUMBASES) INIT((TOTNUMBASES)0),
        SHIPQPT(0:TOTNUMSHIPS),
        DACQPT(TOTNUMAC),
        ACFAILQFPT(0:TOTNUMAC),
        ACFAILQBPT(TOTNUMAC),

        /*BACKORDER LEVEL AT EACH BASE                                */
        BACKORDERS(TOTNUMBASES) INIT((TOTNUMBASES)0),

        /*RANDOM VARIATE SEEDS FOR GENERATION OF FAILURES FOR EACH BASE */
        RVS(TOTNUMBASES),

        /*SHIPNUMBER OF EACH BASE...0 IF NON-CARRIER.                 */
        SHIPNUM(TOTNUMBASES),
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                         /*TRAVEL TIME TO BASE I FROM BASE J                          */
                         TRAVTIME(TOTNUMBASES,TOTNUMBASES),

                         /*TIME IN AND OUT OF PORT FOR EACH SHIP                      */
                         SEASCHED(III:TOTNUMSHIPS,III:II2),
                         PORTSCHED(III:TOTNUMSHIPS,III:II2),
                         IND(III:TOTNUMSHIPS) INIT((TOTNUMSHIPS)1),

                         /*THE HORIZON OF A SHIP'S PORT AND DOCK SCHEDULE             */
                         SCHED_LENGTH(III:TOTNUMSHIPS),

                         /*INDICATES A RANDOM OR DETERMINISTIC SHIPSCHEDULE (0 OR 1)  */
                         SCHED_IND(III:TOTNUMSHIPS),

                         /*IDENTIFIES SHIPS WITH A BASENUMBER                         */
                         SHIPBASE(III:TOTNUMSHIPS),

                         /*TIME OF NEXT DOCK OR DEPLOYMENT ACTIVITY AT EACH BASE.     */
                         SHIPTIME(0:TOTNUMSHIPS),

                         /*THE BASENUMBER OF A BASE'S PORT...0 OF NOT A SHIP          */
                         PORT(TOTNUMBASES),

                         /*NUMBER OF ENGINES PER EACH AIRCRAFT TYPE..                 */
                         ENGCAP(TOTNUMTYPEAC),

                         /*NEXT TIME AN A/C WILL FAIL GIVEN IT IS NOT IN A DOWNED STATE. */
                         FAILTIME(0:TOTNUMAC) INIT(0),

                         /*THE BASE TO WHICH AN AIRCRAFT BELONGS.                     */
                         BASEQ(TOTNUMAC),

                         /*COMBINATION CODE OF EACH TYPE OF AIRCRAFT  (ORDER NUMBER OF */
                         /*INPUT IS ASSIGNED TO ACTUAL CODE).                         */
                         CC(TOTNUMTYPEAC),

                         /*TYPE CODE NUMBER OF EACH A/C                               */
                         TYPE(TOTNUMAC),

                         /*THE LIFE OF ENGINE I ON AIRCRAFT J                         */
                         LIFENG(MAXENGCAP,TOTNUMAC),

                         /*FLYING HOURS PER MONTH FOR A/C TYPE I AT BASE J.           */
                         FOHM(TOTNUMTYPEAC,TOTNUMBASES),

                         /*MEANTIME BETWEEN REMOVALS (LESS REPAIRTIME) FOR EACH CC.   */
                         MTBR(TOTNUMCC),

                         /*THE AMOUNT OF ADJUSTED STOCK AT A BASE (INPUT)             */
                         CAPACITY(TOTNUMBASES)) BIN FLOAT;

  19     2          2    DECLARE(
                         /*TIME WHICH A SYSTEM TAKES TO REPAIR A JTH DEGREE FAILURE.  */
                         REPAIRTIME(-1:3),
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                              /*MEAN PORT AND SEA TIMES FOR EACH SHIP (NORMALLY DISTRIBUTED). */
                              MPORTIME(III:TOTNUMSHIPS),
                              MSEATIME(III:TOTNUMSHIPS),

                              /*TOTAL NUMBER OF OUTSTANDING ENGINES WHICH A SHIP EXPECTS      */
                              /*TO BE RETURNED AT PORT                                        */
                              NUMSENTENG(III:TOTNUMSHIPS) INIT((TOTNUMSHIPS)0),

                              /*MOST RECENT NUMBER OF ENGINES SENT TO PORT.                   */
                              LNUMSENTENG(III:TOTNUMSHIPS) INIT((TOTNUMSHIPS)0)
                              ) BIN FLOAT;

                              /*OTHER VARIABLES.                                              */
 20    2          2           DCL(
                              ACTPORTNUM(TOTNUMBASES),
                              MEANDOCKTIME,DUMMYTIME,TEMP2,
                              SYSPERF,SLACKPERF,
                              SUMWEIGHT INIT(0),
                              BASEFAILEDAC,BOOMT,FAILDEGREE,FAILEDAC,FACS,
                              U1,ACFAILQBOT,ACFAILQDUM,BASENUM,CLOCK,SI,
                              HALT,SAMPTIME,I,J,K,MDFH,NDUM,NUMACTYP,NUMTYPE,RV1,RV2,U,
                              CUMFLYTIME,DAYSTILFAIL,DUM,FLYHRS,FLYTIME,HOLD,HOLD1,INSRT,
                              LINSRT,SWITCH,NUMSAVE,NUMENG,TIMLIM,ACTCODE,ACTNUM,ENGAVAILQBOT,
                              KNT,ENGAVAILQMT,STOCKQMT,RV3,KNT2,RV4,I1,UPA,ENGNEXT,KAP,BASEAVAIL,
                              BOOHOLD,ENGAVAILNUM,ENGAVAILQDUM,ENGAVAILQHOLD,SA,TEMP,
                              RUNIN,NUMSAMPS,STIME,ACFAILTIME,
                              MAXTIME INIT(0),INT1,INT2,
                              DFR(TOTNUMBASES) INIT((TOTNUMBASES)0),
                              MAXSTOCK,MAXAC,
                              TEMP1,SUMDAC,TOTAC,
                              (TOTDFR,MAXTRAV) INIT(0),
                              NJUNAVAILTIME INIT(99999999),
                              L INIT(0), NJ INIT(0)) BIN FLOAT;
 21    2          2           DCL (BASENAME(TOTNUMBASES),STATE(III:TOTNUMSHIPS)) CHAR(20) VAR;
 22    2          2           DCL (ACNAME(TOTNUMTYPEAC)) CHAR(7) VAR;
 23    2          2           DCL DUMMY CHAR(3) VAR;

                              /*INITIALIZATION OF QUEUES AND SYSTEM PARAMETERS.               */
 24    2          2           U=.071653001;
 25    2          2           U1=.040552001;
 26    2          2           RV1=.125542783;
 27    2          2           RV2=.336658007;
 28    2          2           RV3=.014256339;
 29    2          2           RV4=.650125475;
 30    2          2           CLOCK=0;
 31    2          2           BOOMT=1; BOOTOP=0;
 33    2          2           DACOTOP=0;
 34    2          2           ACFAILQFPT(0)=0; ACFAILQBPT=0; ACFAILQBOT=0;
 37    2          2           FAILTIME(0)=-1;
 38    2          2           GET LIST (RVS);
 39    2          2           FACS=SQRT((TOTSAMPLES-2)/(TOTSAMPLES**2-1));
 40    2          2           NUMSAMPS=1;
 41    2          2           GET LIST(SI,LIMTEST,CRIT_BASE,NORMAL_STAT,ALPHANUM,TSTAT,ALPHAT);
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

  42    2          2        IF CRIT_BASE=0 THEN PERCPASS=1-ALPHANDAM; ELSE PERCPASS=1;
  45    2          2        GET LIST(CC_CODE);

                            /*INPUT SHIP MEANS AND STANDARD DEVIATIONS.                    */
                            /*INPUT MEAN TIME BETWEEN REMOVALS AND GENERATE INITIAL DOWNED */
                            /*AIRCRAFT QUEUE.                                             */
                            /*INPUT PORT SCHEDULE AND SEASCHEDULE FOR SHIPS.               */
                            /*INITIALIZE SHIPS RANDOMLY AT PORT OR AT SEA AND SET THEIR NEXT*/
                            /*TIME OF ACTIVITY.                                           */
                            /*QUEUE THE SHIPS IN ORDER OF NEXT ACTIVITY.                  */

  46    2          2        DO KNT=1 TO TOTNUMTYPEAC;
  47    2     1    2           GET LIST (ACNAME(KNT),ENGCAP(KNT),CC(KNT));
  48    2     1    2        END;
  49    2          2        DO KNT=1 TO TOTNUMSHIPS;
  50    2     1    2           GET LIST (MPORTIME(KNT),MSEATIME(KNT),STD_DEV(KNT),SCHED_IND(KNT)
                               ,SCHED_LENGTH(KNT));
  51    2     1    2           DO KNT2=1 TO SCHED_LENGTH(KNT);
  52    2     2    2              IF SCHED_IND(KNT)=1 THEN GET LIST (PORTSCHED(KNT,KNT2),
                                    SEASCHED(KNT,KNT2));
  54    2     2    2              ELSE DO;
  55    2     3    2              PORTSCHED(KNT,KNT2)=SHIPTIME_RNG(MPORTIME(KNT),STD_DEV(KNT),
                                   RVS(KNT));
  56    2     3    2              SEASCHED(KNT,KNT2)=SHIPTIME_RNG(MSEATIME(KNT),STD_DEV(KNT),
                                  RVS(KNT));
  57    2     3    2              END;
  58    2     2    2           END;
  59    2     1    2        END;
  60    2          2        DO KNT=1 TO TOTNUMCC;
  61    2     1    2           GET LIST (MTBR(KNT));
  62    2     1    2        END;
  63    2          2        DO KNT=1 TO TOTNUMAC;
  64    2     1    2            DACQPT(KNT)=KNT+1;
  65    2     1    2        END;
  66    2          2        SHIPTIME=-1; SHIPQPT=0;
  68    2          2        DO KNT=1 TO TOTNUMSHIPS;
  69    2     1    2            SHIPQPT(KNT)=KNT+1;
  70    2     1    2        RV3=RAND(RV3); RV4=RAND(RV4);
  72    2     1    2        IF SCHED_IND(KNT)=1 THEN
  73    2     1    2           DO;
  74    2     2    2              DO KNT2=1 TO SCHED_LENGTH(KNT);
  75    2     3    2                 MPORTIME(KNT)=MPORTIME(KNT)+PORTSCHED(KNT,KNT2);
  76    2     3    2                 MSEATIME(KNT)=MSEATIME(KNT)+SEASCHED(KNT,KNT2);
  77    2     3    2              END;
  78    2     2    2              MPORTIME(KNT)=MPORTIME(KNT)/SCHED_LENGTH(KNT);
  79    2     2    2              MSEATIME(KNT)=MSEATIME(KNT)/SCHED_LENGTH(KNT);
  80    2     2    2           END;
  81    2     1    2            IF RV3>MSEATIME(KNT)/(MSEATIME(KNT)+MPORTIME(KNT))
  82    2     1    2                THEN DO;
  83    2     2    2                    STATE(KNT)='AT PORT';
  84    2     2    2                 IF SCHED_IND(KNT)=0 THEN
  85    2     2    2                 SHIPTIME(KNT)=RV4*SHIPTIME_RNG(MPORTIME(KNT),STD_DEV(KNT),
                                       RVS(KNT));
  86    2     2    2                 ELSE
                                          DO;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

  87    2     3     2                       SHIPTIME(KNT)=RV4*PORTSCHED(KNT,1);
  88    2     3     2                       IND(KNT)=2;
  89    2     3     2                    END;
  90    2     2     2                END;
  91    2     1     2                ELSE DO;
  92    2     2     2                    STATE(KNT)='AT SEA';
  93    2     2     2                 IF SCHED_IND(KNT)=0 THEN
  94    2     2     2                 SHIPTIME(KNT)=RV4*SHIPTIME_RNG(MSEATIME(KNT),STD_DEV(KNT),
                                        RVS(KNT));
  95    2     2     2                 ELSE SHIPTIME(KNT)=RV4*SEASCHED(KNT,1);
  96    2     2     2                END;
  97    2     1     2            DUM=SHIPQPT(0);
  98    2     1     2            DO WHILE (SHIPTIME(DUM)>SHIPTIME(KNT));
  99    2     2     2                DUM=SHIPQPT(DUM);
 100    2     2     2            END;
 101    2     1     2            HOLD=SHIPQPT(DUM);
 102    2     1     2            SHIPQPT(DUM)=KNT;
 103    2     1     2            SHIPQPT(KNT)=HOLD;
 104    2     1     2        END;

                             /*INPUT THE DISTRIBUTION FOR FAILURE TYPES.                */
                             /*DISTRIBUTION OF DATA IS IN DENSITY FORM                  */
                             /*INPUT REPAIRTIMES.                                       */

 105    2           2        DO KNT=1 TO TOTNUMCC;
 106    2     1     2        GET LIST (CCDISTDENS(KNT,3),CCDISTDENS(KNT,2),CCDISTDENS(KNT,1),
                             CCDISTDENS(KNT,-1),CCDISTDENS(KNT,0));
 107    2     1     2        CCDIST(KNT,3)=CCDISTDENS(KNT,3);  CCDIST(KNT,-1)=1;
 109    2     1     2           DO INDI=2 TO 0 BY -1;
 110    2     2     2           CCDIST(KNT,INDI)=CCDIST(KNT,INDI+1)+CCDISTDENS(KNT,INDI);
 111    2     2     2        END;
 112    2     1     2        END;
 113    2           2        GET LIST(REPAIRTIME(3),REPAIRTIME(2),REPAIRTIME(1),REPAIRTIME(-1),
                                      REPAIRTIME(0));
 114    2           2        PUT EDIT('SYSTEM STRUCTURE FOR DEPOT= ',DEPOT,' OF ENGINE TYPE-'||
                             'MODEL ',ENGMODEL) (COL(27),A,A,A,A);
 115    2           2        IF TOTNUMSHIPS=0 THEN PUT SKIP EDIT('(CONTINUOUS PIPELINE)')
                             (COL(47),A);
 117    2           2        ELSE PUT SKIP EDIT('(TWISTED PIPELINE)') (COL(48),A);
 118    2           2           DO I=1 TO TOTNUMCC;
 119    2     1     2              PUT SKIP(2) EDIT('CC= ',CC_CODE(I),'MTBR(INC. INSP)= ',
                                                   MTBR(I)) (COL(15),A,F(3),COL(25),A,F(5,2));
 120    2     1     2              PUT SKIP EDIT('MAINT LEVEL: ','3RD-DEG','2ND-DEG','1ST-DEG',
                                           '1ST-DEG/OH','OVERHAUL')(COL(18),A,COL(38),A,
                                           COL(50),A,COL(62),A,COL(74),A,COL(89),A);
 121    2     1     2              PUT SKIP EDIT('PROB MAINT:') (COL(18),A);
 122    2     1     2              M=39;
 123    2     1     2                 DO N=3 TO -1 BY -1;
 124    2     2     2                    IF N=0 THEN PUT EDIT(CCDISTDENS(I,-1))
                                                             (COL(77),F(5,3));
 126    2     2     2                    ELSE IF N=-1 THEN PUT EDIT(CCDISTDENS(I,0))
                                                                  (COL(91),F(5,3));
 128    2     2     2                    ELSE PUT EDIT(CCDISTDENS(I,N)) (COL(M),F(5,3));
 129    2     2     2                    M=M+12;
 130    2     2     2                 END;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 131    2    1      2              PUT SKIP EDIT('REPAIRTIME:') (COL(19),A);
 132    2    1      2              M=39;
 133    2    1      2                 DO N=3 TO -1 BY -1;
 134    2    2      2                    IF N=0 THEN PUT EDIT(REPAIRTIME(-1))(COL(77),F(5,2));
 136    2    2      2                    ELSE IF N=-1 THEN PUT EDIT(REPAIRTIME(0))
                                         (COL(91),F(5,2));
 138    2    2      2                    ELSE PUT EDIT(REPAIRTIME(N)) (COL(M),F(5,2));
 139    2    2      2                    M=M+12;
 140    2    2      2                 END;
 141    2    1      2           END;
 142    2           2        PUT SKIP(3) EDIT('-') (A);
 143    2           2           DO I=2 TO 120;
 144    2    1      2              PUT EDIT('-') (A);
 145    2    1      2           END;
 146    2           2        L=0; NJ=0;

                             /*IN PUT BASE ATTRIBUTES                                            */
                             /*   ASSIGN AIRCRAFT TO BASES, GENERATE AND QUEUE THE INITIAL      */
                             /*AIRCRAFT FAILURES, AND ASSUME NO INITIAL DOWNED AC.              */
                             /*   CALCULATE A RUNIN TIME AND HALT TIME.                         */

 148    2           2        DO I=1 TO TOTNUMBASES;
 149    2    1      2        RV1=RVS(I);
 150    2    1      2        GET LIST(BASENAME(I),BASENUM,SHIPNUM(I),ACTPORTNUM(I),
                             NUMACTYP,PORT(I),PLIND(I),PLSTOCK(I),CRITICAL(I),
                             DES_CRIT(I),WEIGHT(I));
 151    2    1      2        SUMWEIGHT=SUMWEIGHT+WEIGHT(I);
 152    2    1      2            DO I1=3 TO -1 BY -1;
 153    2    2      2               GET LIST (NEXTECH(I,I1));
 154    2    2      2            END;
 155    2    1      2            DO I1=1 TO TOTNUMBASES;
 156    2    2      2               GET LIST (TRAVTIME(I,I1));
 157    2    2      2               IF MAXTRAV<TRAVTIME(I,I1) THEN MAXTRAV=TRAVTIME(I,I1);
 159    2    2      2            END;
 160    2    1      2           IF SHIPNUM(I)¬=0 THEN
 161    2    1      2              DO;
 162    2    2      2              SHIPBASE(SHIPNUM(I))=I;
 163    2    2      2              IF STATE(SHIPNUM(I))='AT PORT' THEN
 164    2    2      2                 DO;
 165    2    3      2                    LSTIME(I)=SHIPTIME(SHIPNUM(I));
 166    2    3      2                    LBOTIME(I)=SHIPTIME(SHIPNUM(I));
 167    2    3      2                    LOACTIME(I)=SHIPTIME(SHIPNUM(I));
 168    2    3      2                 END;
 169    2    2      2              END;
 170    2    1      2        IF MOD(I,3)=0 & I¬=TOTNUMBASES THEN
 171    2    1      2           DO;
 172    2    2      2              PUT PAGE;
 173    2    2      2              PUT SKIP EDIT ('-') (A);
 174    2    2      2              DO N=1 TO 119;
 175    2    3      2                 PUT EDIT ('-') (A);
 176    2    3      2              END;
 177    2    2      2           END;
 178    2    1      2           PUT SKIP(3) EDIT('BASE NAME: ',BASENAME(I)) (COL(4),A,A);
 179    2    1      2           PUT SKIP EDIT('BASE #: ',I) (COL(30),A,F(2));
 180    2    1      2           IF SHIPNUM(I)>0 THEN PUT EDIT('MEAN SEA TIME- ',MSEATIME(
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                  SHIPNUM(I))) (COL(83),A,F(5,2));
 182    2    1    2               PUT SKIP EDIT('CARRIER #: ',SHIPNUM(I)) (COL(30),A,F(2));
 183    2    1    2               IF SHIPNUM(I)>0 THEN PUT EDIT('MEAN DOCK TIME= ',MPORTIME(
                                  SHIPNUM(I))) (COL(83),A,F(5,2));
 185    2    1    2               IF SHIPNUM(I)>0 THEN PUT SKIP EDIT('CARRIER''S PORT: BASE #',
                                  PORT(I)) (COL(30),A,F(2));
 187    2    1    2               ELSE
                                     DO;
 188    2    2    2                     PUT SKIP EDIT('ACTING PORT #: ',ACTPORTNUM(I))
                                        (COL(30),A,F(2));
 189    2    2    2                  END;
 190    2    1    2               IF NUMACTYP=0 THEN DUMMY='NO';
 192    2    1    2               ELSE DUMMY='YES';
 193    2    1    2               SWITCH=1;
 194    2    1    2                  DO WHILE(SWITCH=1);
 195    2    2    2                     IF HD(I)=4 THEN SWITCH=0;
 197    2    2    2                     ELSE IF I=NEXTECH(I,HD(I)) THEN SWITCH=0;
 199    2    2    2                     ELSE HD(I)=HD(I)+1;
 200    2    2    2                  END;
 201    2    1    2            PUT SKIP(2) EDIT('HIGHEST REPAIR DEGREE= ',
                                           HD(I),'FLYING ACTIVITY?- ',DUMMY)
                                           (COL(34),A,F(1),COL(67),A,A);
 202    2    1    2            IF ACTPORTNUM(I)>0 & NUMACTYP=0 THEN
 203    2    1    2               DO;
 204    2    2    2                  PUT SKIP(2) EDIT('|') (X(61),A);
 205    2    2    2               PUT EDIT('- - - - M A I N T E N A N C E   A C T I V I T I E S' ||
                                        '- - - -') (A);
 206    2    2    2                  PUT SKIP(1) EDIT('|MAINT LEVEL:','3RD-DEG','2ND-DEG',
                                     '1ST-DEG','1ST-DEG/CH','OVERHAUL')(COL(62),A,COL(78),A,
                                     COL(86),A,COL(94),A,COL(102),A,COL(113),A);
 207    2    2    2               PUT SKIP EDIT('|BASE #:') (COL(62),A);
 208    2    2    2               M=80;
 209    2    2    2                  DO N=3 TO -1 BY -1;
 210    2    3    2                     IF N=0 THEN PUT EDIT(NEXTECH(I,-1)) (COL(104),F(2));
 212    2    3    2                     ELSE IF N=-1 THEN PUT EDIT(NEXTECH(I,0)) (COL(115),F(2));
 214    2    3    2                     ELSE PUT EDIT(NEXTECH(I,N)) (COL(M),F(2));
 215    2    3    2                     M=M+8;
 216    2    3    2                  END;
 217    2    2    2                  PUT SKIP EDIT('|TRAVEL TIME:')(COL(62),A);
 218    2    2    2                        M=80;
 219    2    2    2                           DO N=3 TO -1 BY -1;
 220    2    3    2                        IF NEXTECH(I,N)¬=0 THEN DO;
 222    2    4    2                           IF N=0 THEN PUT EDIT(TRAVTIME(I,NEXTECH(I,-1)))
                                                               (COL(104),F(2));
 224    2    4    2                             ELSE IF N=-1 THEN PUT EDIT
                                                     (TRAVTIME(I,NEXTECH(I,0))) (COL(115),F(2));
 226    2    4    2                             ELSE PUT EDIT(TRAVTIME(I,NEXTECH(I,N))) (COL(M),
                                                                                         F(2));
 227    2    4    2                        END;
 228    2    3    2                             ELSE DO;
 229    2    4    2                                IF N=0 THEN PUT EDIT ('N/A') (COL(104),A);
 231    2    4    2                                ELSE IF N=-1 THEN PUT EDIT ('N/A') (COL(115),A);
 233    2    4    2                                   ELSE IF N=3 THEN PUT EDIT ('N/A') (COL(80),
                                                         A);
 235    2    4    2                                      ELSE IF N=2 THEN PUT EDIT ('N/A')(COL(88)
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                           ,A);
 237    2    4    2                            ELSE PUT EDIT ('N/A') (COL(96),A);
 238    2    4    2                   END;
 239    2    3    2                   M=M+8;
 240    2    3    2                 END;
 241    2    2    2         END;
 242    2    1    2         IF NUMACTYP¬=0 THEN DO;
 244    2    2    2      PUT SKIP(2) EDIT('- - - - - -A I R C R A F T  D E T A I L S- '||
                                '- - - - - - - - -|') (COL(4),A);
 245    2    2    2         PUT EDIT('- - - - M A I N T E N A N C E  A C T I V I T I E S' ||
                             '- - - - -') (A);
 246    2    2    2         PUT SKIP(1) EDIT('TYPE','CC','UPA','FH/MO/ACFT','# ACFT',
                                      '|MAINT LEVEL:','3RD-DEG','2ND-DEG','1ST-DEG',
                                      '1ST-DEG/OH','OVERHAUL') (COL(8),A,COL(20),A,
                                      COL(27),A,COL(35),A,COL(50),A,COL(62),A,COL(78),
                                      A,COL(86),A,COL(94),A,COL(102),A,COL(113),A);
 247    2    2    2         PUT SKIP EDIT('|BASE #:') (COL(62),A);
 248    2    2    2         M=80;
 249    2    2    2            DO N=3 TO -1 BY -1;
 250    2    3    2               IF N=0 THEN PUT EDIT(NEXTECH(I,-1)) (COL(104),F(2));
 252    2    3    2               ELSE IF N=-1 THEN PUT EDIT(NEXTECH(I,0)) (COL(115),F(2));
 254    2    3    2               ELSE PUT EDIT(NEXTECH(I,N)) (COL(M),F(2));
 255    2    3    2               M=M+8;
 256    2    3    2            END;
 257    2    2    2         SWITCH=1;
 258    2    2    2         END;
 259    2    1    2         ELSE
                               DO;
 260    2    2    2            PUT SKIP(2) EDIT('-') (A);
 261    2    2    2               DO N=2 TO 120;
 262    2    3    2                  PUT EDIT('-') (A);
 263    2    3    2               END;
 264    2    2    2            END;
 265    2    1    2             IF NUMACTYP>0 THEN TOTNUMACB=TOTNUMACB+1;
 267    2    1    2          DO J=1 TO NUMACTYP;
 268    2    2    2              GET LIST (ACTCODE,FOHM(ACTCODE,I),NUMTYPE);
 269    2    2    2              P(I,*)=FOHM(ACTCODE,I)*ENGCAP(ACTCODE)*CCDISTDENS(
                                 CC(ACTCODE),*)*NUMTYPE/(30*MTBR(CC(ACTCODE)))+P(I,*);
 270    2    2    2              IF SHIPNUM(I)>0 THEN FOHM(ACTCODE,I)=
                                    ((MSEATIME(SHIPNUM(I))+MPORTIME(SHIPNUM(I)))/
                                   MSEATIME(SHIPNUM(I)))*FOHM(ACTCODE,I);
 272    2    2    2              AIRCRAFT(I)=AIRCRAFT(I)+NUMTYPE;
 273    2    2    2              ENGINES(I)=ENGINES(I)+ENGCAP(ACTCODE)*NUMTYPE;
 274    2    2    2            PUT SKIP EDIT(ACNAME(ACTCODE),CC_CODE(CC(ACTCODE)),
                                      ENGCAP(ACTCODE),
                                      FOHM(ACTCODE,I),NUMTYPE,'|') (COL(8),A,COL(19),
                                     F(3),COL(27),F(2),COL(37),F(6,2),COL(51),
                                     F(5,2),COL(62),A);
 275    2    2    2            IF SWITCH=1 THEN
 276    2    2    2               DO;
 277    2    3    2                  PUT EDIT('TRAVEL TIME:') (A);
 278    2    3    2                  M=80;
 279    2    3    2                     DO N=3 TO -1 BY -1;
 280    2    4    2                       IF N=0 THEN PUT EDIT(TRAVTIME(I,NEXTECH(I,-1)))
                                                  (COL(104),F(2));
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 292    2    4     2                      ELSE IF N=-1 THEN PUT EDIT
                                                (TRAVTIME(I,NEXTECH(I,0))) (COL(115),F(2));
 284    2    4     2                      ELSE PUT EDIT(TRAVTIME(I,NEXTECH(I,N))) (COL(M),
                                                                                   F(2));
 285    2    4     2                      M=M+8;
 286    2    4     2                   END;
 287    2    3     2                SWITCH=0:
 288    2    3     2             END;
 289    2    2     2          IF J=NUMACTYP THEN
 290    2    2     2             DO;
 291    2    3     2             PUT SKIP(2) EDIT('-') (A);
 292    2    3     2                   DO N=2 TO 120;
 293    2    4     2                      PUT EDIT('-') (A);
 294    2    4     2                   END;
 295    2    3     2             END;
 296    2    2     2          FOHM(ACTCODE,I)=FOHM(ACTCODE,I)*NUMTYPE/CEIL(NUMTYPE);
 297    2    2     2          NUMTYPE=CEIL(NUMTYPE);
 298    2    2     2            DO K=1 TO NUMTYPE;
 299    2    3     2                L=L+1;
 300    2    3     2             IF FOHM(ACTCODE,I)>0 THEN
 301    2    3     2             DO;
 302    2    4     2                TYPE(L)=ACTCODE;
 303    2    4     2                BASEO(L)=I;
 304    2    4     2                CUMFLYTIME=0; HOLD=0;
 306    2    4     2                FLYHRS=RNG(MTBR(CC(ACTCODE)),RVS(I));
 307    2    4     2                DAYSTILFAIL=FLYHRS*30/FOHM(TYPE(L),I);
 308    2    4     2                IF SHIPNUM(I)¬=0
 309    2    4     2                   THEN DO;
 310    2    5     2                      IF STATE(SHIPNUM(I))='AT PORT'
 311    2    5     2                         THEN DO;
 312    2    6     2                            HOLD=SHIPTIME(SHIPNUM(I));
 313    2    6     2                          DUM=IND(SHIPNUM(I));
 314    2    6     2                           TIMLIM=SEASCHED(SHIPNUM(I),DUM);
 315    2    6     2                         END;
 316    2    5     2                         ELSE DO;
 317    2    6     2                           HOLD=0;
 318    2    6     2                           DUM=IND(SHIPNUM(I));
 319    2    6     2                           TIMLIM=SHIPTIME(SHIPNUM(I));
 320    2    6     2                         END;
 321    2    5     2                       DO WHILE (DAYSTILFAIL>TIMLIM);
 322    2    6     2                         CUMFLYTIME=TIMLIM+CUMFLYTIME +
                                                PORTSCHED(SHIPNUM(I),DUM);
 323    2    6     2                          DUM=MOD(DUM,SCHED_LENGTH(SHIPNUM(I)))+1;
 324    2    6     2                           DAYSTILFAIL=DAYSTILFAIL-TIMLIM;
 325    2    6     2                         TIMLIM=SEASCHED(SHIPNUM(I),DUM);
 326    2    6     2                       END;
 327    2    5     2                END;
 328    2    4     2                FAILTIME(L)=HOLD+CUMFLYTIME+DAYSTILFAIL;
 329    2    4     2          END;
 330    2    3     2          ELSE FAILTIME(L)=3*2**LIMTEST*TOTSAMPLES*SI;
 331    2    3     2                ACFAILQDUM=ACFAILQBOT;
 332    2    3     2                DO WHILE (FAILTIME(L)<FAILTIME(ACFAILQDUM));
 333    2    4     2                   ACFAILQDUM=ACFAILQBPT(ACFAILQDUM);
 334    2    4     2                END;
 335    2    3     2                IF ACFAILQDUM=ACFAILQBOT
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 336    2    3    2                       THEN DO;
 337    2    4    2                           ACFAILQFPT(ACFAILQBOT)=L;
 338    2    4    2                           ACFAILQFPT(L)=L;
 339    2    4    2                           ACFAILQBPT(L)=ACFAILQBOT;
 340    2    4    2                           ACFAILQBOT=L;
 341    2    4    2                       END;
 342    2    3    2                       ELSE DO;
 343    2    4    2                           ACFAILQFPT(L)=ACFAILQFPT(ACFAILQDUM);
 344    2    4    2                           ACFAILQFPT(ACFAILQDUM)=L;
 345    2    4    2                           ACFAILQBPT(L)=ACFAILQDUM;
 346    2    4    2                           ACFAILQBPT(ACFAILQFPT(L))=L;
 347    2    4    2                       END;
 348    2    3    2                       LIFENG(1,L)=0;
 349    2    3    2                       NJ=NJ+1;
 350    2    3    2                   DO ENGNEXT=2 TO ENGCAP(ACTCODE);
 351    2    4    2                       RV3=RAND(RV3);
 352    2    4    2                       LIFENG(ENGNEXT,L)=RV3*MTBR(CC(ACTCODE));
 353    2    4    2                       NJ=NJ+1;
 354    2    4    2                   END;
 355    2    3    2                END;
 356    2    2    2             END;
 357    2    1    2          END;
 358    2         2          DO I=1 TO TOTNUMBASES;
 359    2    1    2             IF ACTPORTNUM(I)>0 THEN
 360    2    1    2                DO;
 361    2    2    2                   DO K=1 TO TOTNUMBASES;
 362    2    3    2                      DO J=-1 TO 3;
 363    2    4    2                         IF NEXTECH(K,J)=I & I¬=K THEN
 364    2    4    2                         P(I,J)=P(I,J)+P(K,J);
 365    2    4    2                      END;
 366    2    3    2                   END;
 367    2    2    2             END;
 368    2    1    2          END;
 369    2         2          MEANDOCKTIME=0;
 370    2         2          DO I=1 TO TOTNUMSHIPS;
 371    2    1    2          MEANDOCKTIME=MEANDOCKTIME+MPORTIME(I);
 372    2    1    2          END;
 373    2         2          IF TOTNUMSHIPS>0 THEN MEANDOCKTIME=MEANDOCKTIME/TOTNUMSHIPS;
 375    2         2          PUT PAGE;
 376    2         2          ACFAILTIME=FAILTIME(ACFAILQFPT(0));
 377    2         2             DO I=1 TO TOTNUMBASES;
 378    2    1    2                DO J=-1 TO 3;
 379    2    2    2                   DFR(I)=DFR(I)+P(I,J);
 380    2    2    2                END;
 381    2    1    2             END;
 382    2         2             DO I=1 TO TOTNUMBASES;
 383    2    1    2             IF AIRCRAFT(I)>0 & DFR(I)>0 THEN IF MAXTIME<ENGINES(I)/DFR(I)
                                 THEN
 385    2    1    2                MAXTIME=ENGINES(I)/DFR(I);
 386    2    1    2                TOTDFR=TOTDFR+DFR(I);
 387    2    1    2             END;
 388    2         2          RUNIN=CEIL(MAXTIME);
 389    2         2          SAMPTIME=RUNIN;
 390    2         2          HALT=RUNIN+TOTSAMPLES*SI;
 391    2         2          SHIPTIME(0)=2**LIMTEST*HALT+1;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 392   2          2             SPARES=FLOOR(2*TOTDFR*(REPAIRTIME(0)+2*MAXTRAV));
 393   2          2            B2:BEGIN;
 394   3          3             DCL(
                                /*QUEUE MECHANISM FOR ENGINE AVAILABILITY                    */
                                ENGAVAILTIME(0:SPARES),
                                ENGAVAILQFPT(0:SPARES),
                                ENGAVAILQBPT(0:SPARES),
                                BASENGAVAIL(SPARES)) BIN FLOAT;

 395   3          3             DCL AVAILQ ENTRY(BIN FLOAT,BIN FLOAT,BIN FLOAT,BIN FLOAT);
 396   3          3             ENGAVAILQFPT(0)=0;
 397   3          3             ENGAVAILQBPT=0;
 398   3          3             ENGAVAILQBOT=0;
 399   3          3             ENGAVAILQMT=1;
 400   3          3             ENGAVAILTIME(0)=-1;
 401   3          3             DO KNT=1 TO SPARES;
 402   3    1     3                ENGAVAILQFPT(KNT)=KNT+1;
 403   3    1     3             END;

                                /*CALCULATE PIPELINE AND SAFETY STOCK LEVELS AND MAKE PIEPLINE  */
                                /*STOCK AVAILABLE AT ITS DESTINATION IN THE FUTURE (INITIALIZE  */
                                /*THE PIPELINE).                                                */

 404   3          3             DO I=1 TO TOTNUMBASES;
 405   3    1     3                DO J=-1 TO 3;
 406   3    2     3                   DO K=1 TO TOTNUMBASES;
 407   3    3     3                      IF NEXTECH(K,J)=I THEN
 408   3    3     3                         DO;
 409   3    4     3                            IF SHIPNUM(K)=0 | I=K THEN
 410   3    4     3                               DO;
 411   3    5     3                                  IF HD(I)>J THEN
 412   3    5     3                                     DO;
 413   3    6     3                                        TEMP=P(K,J)*TRAVTIME(K,I);
 414   3    6     3                                        R(I)=R(I)+TEMP;
 415   3    6     3                                        TEMP1=TRAVTIME(K,I);
 416   3    6     3                                     END;
 417   3    5     3                                     ELSE IF ACTPORTNUM(K)>0 & TOTNUMSHIPS>0 THEN
 418   3    5     3                                     DO;
 419   3    6     3                                        TEMP=P(K,J)*MAX(0,REPAIRTIME(J)+TRAVTIME
                                                           (K,I)-MEANDOCKTIME);
 420   3    6     3                                        R(I)=R(I)+TEMP;
 421   3    6     3                                        TEMP1=REPAIRTIME(J);
 422   3    6     3                                     END;
 423   3    5     3                                  ELSE
                                                        DO;
 424   3    6     3                                        TEMP=P(K,J)*(TRAVTIME(K,I)+REPAIRTIME(J)
                                                           );
 425   3    6     3                                        R(I)=R(I)+TEMP;
 426   3    6     3                                        TEMP1=TRAVTIME(K,I)+REPAIRTIME(J);
 427   3    6     3                                     END;
 428   3    5     3                                  IF MAXPL(I)<TEMP1 THEN MAXPL(I)=TEMP1;
 430   3    5     3                               END;
 431   3    4     3                            ELSE IF HD(I)<=J THEN
 432   3    4     3                               DO;
 433   3    5     3                                  TEMP=P(K,J)*MAX(0,REPAIRTIME(J)-MPORTIME(
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                              SHIPNUM(K)));
 434    3    5     3                                R(I)=R(I)+TEMP;
 435    3    5     3                          TEMP1=REPAIRTIME(J);
 436    3    5     3                          IF MAXPL(I)<TEMP1 THEN MAXPL(I)=TEMP1;
 438    3    5     3                       END;
 439    3    4     3                 END;
 440    3    3     3              IF NEXTECH(I,J)=K THEN
 441    3    3     3                 DO;
 442    3    4     3                    IF SHIPNUM(I)=0 | I=K THEN
 443    3    4     3                       DO;
 444    3    5     3                          IF ACTPORTNUM(I)>0 & TOTNUMSHIPS>0 THEN
 445    3    5     3                             DO;
 446    3    6     3                                TEMP=P(I,J)*MAX(0,TRAVTIME(K,I)-
                                                    MEANDOCKTIME);
 447    3    6     3                                R(I)=R(I)+TEMP;
 448    3    6     3                                TEMP1=TRAVTIME(K,I);
 449    3    6     3                             END;
 450    3    5     3                          ELSE
                                                 DO;
 451    3    6     3                                TEMP=P(I,J)*TRAVTIME(K,I);
 452    3    6     3                                R(I)=R(I)+TEMP;
 453    3    6     3                                TEMP1=TRAVTIME(K,I);
 454    3    6     3                             END;
 455    3    5     3                          IF MAXPL(I)<TEMP1 THEN MAXPL(I)=TEMP1;
 457    3    5     3                       END;
 458    3    4     3                    ELSE
                                           DO;
 459    3    5     3                          TEMP=P(I,J)*MSEATIME(SHIPNUM(I));
 460    3    5     3                                R(I)=R(I)+TEMP;
 461    3    5     3                       END;
 462    3    4     3                 END;
 463    3    3     3           END;
 464    3    2     3        END;
 465    3    1     3     END;
 466    3          3     CAPACITY=0;
 467    3          3     DO I=1 TO TOTNUMBASES;
 469    3    1     3        IF PLIND(I)=0 THEN PLSTOCK(I)=SAFETY_STOCK(R(I),CRITICAL(I));
 470    3    1     3        R(I)=PLSTOCK(I)-MAX(0,PLSTOCK(I)-FLOOR(R(I)+.5));
 471    3    1     3        CAPACITY(I)=PLSTOCK(I)-R(I);
 472    3    1     3              IF SHIPNUM(I)>0 THEN
 473    3    1     3                 DO;
 474    3    2     3                       NUMSENTENG(SHIPNUM(I))=R(I);
 475    3    2     3                       STOCKLEVEL(PORT(I))=STOCKLEVEL(PORT(I))+R(I);
 476    3    2     3                 END;
 477    3    1     3              ELSE
                                    DO CH=1 TO R(I);
 478    3    2     3                    RVS(I)=RAND(RVS(I));
 479    3    2     3                    TEMP1=RVS(I)*MAXPL(I);
 480    3    2     3                    CALL AVAILC(I,I,0,TEMP1);
 481    3    2     3                 END;
 482    3    1     3     END;
 483    3          3     PUT SKIP (5) EDIT ('*') (A);
 484    3          3     DO N=1 TO 49;
 485    3    1     3        PUT EDIT ('*') (A);
 486    3    1     3     END;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 487    3          3           PUT EDIT(' SPARE STOCK LEVELS ') (A);
 488    3          3           DO N=1 TO 50;
 489    3     1    3              PUT EDIT ('*') (A);
 490    3     1    3           END;
 491    3          3           PUT SKIP EDIT('*','*') (A,COL(120),A);
 492    3          3              PUT SKIP EDIT('*','BASE#:','BASENAME:','LEVEL:','DOD COMPU' ||
                                  'TATION','INPUT','*')(A,COL(28),A,COL(39),A,COL(60),A,COL(71),A,
                                  COL(89),A,COL(120),A);
 493    3          3           PUT SKIP EDIT('*','*') (A,COL(120),A);
 494    3          3           MAXAC=0; MAXSTOCK=0;
 496    3          3              DO I=1 TO TOTNUMBASES;
 497    3     1    3                 TEMP=STOCKLEVEL(I);
 498    3     1    3                 STOCKLEVEL(I)=TEMP+CAPACITY(I);
 499    3     1    3              PUT SKIP EDIT('*',I,BASENAME(I),PLSTOCK(I))(A,COL(30),F(2),
                                  COL(39),A,COL(62),F(2));
 500    3     1    3              IF PLIND(I)=0 THEN PUT EDIT('X','*')(COL(78),A,COL(120),A);
 502    3     1    3              ELSE PUT EDIT('X','*')(COL(91),A,COL(120),A);
 503    3     1    3                 IF MAXSTOCK<STOCKLEVEL(I)+R(I) THEN MAXSTOCK=
                                     STOCKLEVEL(I)+R(I);
 505    3     1    3                 IF MAXAC<AIRCRAFT(I) THEN MAXAC=AIRCRAFT(I);
 507    3     1    3              END;
 508    3          3           PUT SKIP EDIT('*','*') (A,COL(120),A);
 509    3          3           PUT SKIP EDIT('*') (A);
 510    3          3           DO N=1 TO 119;
 511    3     1    3           PUT EDIT ('*') (A);
 512    3     1    3           END;
 513    3          3           PUT SKIP(2) EDIT('(NOTE: AN ''X'' UNDER THE FIRST HEADING DENOTES'||
                               ' THAT THE SPARE STOCKLEVEL HAS BEEN COMPUTED IN THE PROGRAM USING')
                               (A);
 514    3          3           PUT SKIP EDIT('DOD-REQUIREMENTS, WHILE AN ''X'' UNDER THE SECOND'||
                                  ' HEADING DENOTES THAT THE SPARE STOCKLEVEL IS A USER INPUT)')
                                  (COL(9),A);
 515    3          3           PUT PAGE;
 516    3          3          B3:BEGIN;
                               /*QUEUE MECHANISM FOR BACKORDERS                                    */
 517    4          4           DCL (
                               BASEBO(NJ),
                               BOOPT(0:NJ),

                               /*DISTRIBUTION ARRAYS FOR STOCK AND BACKORDER LEVELS.               */
                               STOCKDIST(TOTNUMBASES,0:MAXSTOCK+1)
                               INIT((TOTNUMBASES*(MAXSTOCK+2))0),
                               BODIST(TOTNUMBASES,0:MAXSTOCK+1) INIT((TOTNUMBASES*(MAXSTOCK+2))0),
                               DACDIST(TOTNUMBASES,0:MAXAC) INIT((TOTNUMBASES*(MAXAC+1))0)
                               ) BIN FLOAT;

 518    4          4           DCL FAILGEN ENTRY(BIN FLOAT,BIN FLOAT,BIN FLOAT);
 519    4          4           DCL BACKORDQ ENTRY(BIN FLOAT);
 520    4          4           DO KNT=1 TO NJ;
 521    4     1    4              BOOPT(KNT)=KNT+1;
 522    4     1    4           END;


                               /*   SYNCHRONIZATION LOOP CHOOSES THE MOST IMMEDIATE EVENT AND  */
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                              /*CALLS THE APPROPRIATE FUNCTION WHICH EXECUTES THAT EVENT.        */
                              /*THE CURRENT VALUE OF CLOCK IS THE CURRENT SYSTEM SIMULATION      */
                              /*TIME.                                                            */

 523    4         4         SYNCH:
                              DO WHILE(1=1);
 524    4    1    4           CLOCK=MIN(SHIPTIME(SHIPQPT(0)),ACFAILTIME,NJUNAVAILTIME,
                                    SAMPTIME,HALT);
 525    4    1    4               IF CLOCK=NJUNAVAILTIME THEN CALL ENGAVAILEVENT;
 527    4    1    4             IF CLOCK= ACFAILTIME THEN CALL PLANEFAILEVENT;
 529    4    1    4             IF CLOCK=SHIPTIME(SHIPQPT(0)) THEN CALL SHIPEVENT;
 531    4    1    4             IF CLOCK=SAMPTIME THEN CALL SAMPLE;
 533    4    1    4           END;


                              /*****************************************************************/
                              /*PLANEFAILEVENT:                                                  */
                              /*PLANEFAILEVENT IS CALLED WHEN A FAILURE OCCURS.                  */
                              /*THE EVENT INCLUDES THE FOLLOWING ACTIVITY.                       */
                              /*    1)   REMOVE THE FAILED AIRCRAFT FROM ITS QUEUE.              */
                              /*    2)   DETERMINE THE DEGREE OF FAILURE WHICH OCCURED AND       */
                              /*              WHICH BASE CAN REPAIR IT.                          */
                              /*    3)   ATTEMPT TO MAKE THE PLANE OPERATIONAL                   */
                              /*         IF STOCK EXISTS,                                        */
                              /*              PLACE AN ENGINE ON THE AIRCRAFT.                   */
                              /*              UPDATE DISTRIBUTIONS AND LEVELS.                   */
                              /*              DETERMINE NEXT FAILURE TIME FOR AIRCRAFT AND       */
                              /*              PLACE IT BACK IN THE FAILURE QUEUE.                */
                              /*         IF NOT,                                                 */
                              /*              PLACE THE AIRCRAFT IN A DOWNED AIRCRAFT QUEUE      */
                              /*              RECORD A BACKORDER AND QUEUE IT.                   */
                              /*              UPDATE LEVELS AND DISTRIBUTIONS.                   */
                              /*    4) IF AN ENGINE IS BEING SENT TO A BASE WITH LOWER REPAIR    */
                              /*         DEGREE (IN THE EVENT OF APORT WHICH NEITHER REPAIRS     */
                              /*         NOR FLIES.)                                             */
                              /*         PROVISIONS MUST BE MADE FOR DEMAND ACCOUNTING.          */
                              /*         FOR INSTANCE, A SHIP PLACES A DEMAND ON THE PORT        */
                              /*         WHICH DOESN'T FLY OR REPAIR.  THE PORT WILL THEN SEND   */
                              /*         THE ENGINE TO THE PROPER REPAIR BASE AND DEMAND ONE IN  */
                              /*         RETURN.                                                 */
                              /*    5)   WHEN AN ENGINE TRANSACTION DOES NOT REQUIRE AN INTER-   */
                              /*         MEDIARY PORT, THE ENGINE EXCHANGE TAKES PLACE AS EX-    */
                              /*         PECTED.  THAT IS, THE BASE AT WHICH AN ENGINE FAILS     */
                              /*         EXCHANGES ENGINES WITH THE REPAIR BASE.  IF THE BASE    */
                              /*         ITSELF IS ABLE TO REPAIR THE FAILED ENGINE, NO EX-      */
                              /*         CHANGE IS NECESSARY.IF THE FAILURE OCCURS ON A SHIP     */
                              /*         , AND THE SHIP CAN'T REPAIR IT, THE REPAIR BASE SENDS   */
                              /*         AN  ENGINE TO THE SHIP'S PORT.                          */
                              /*****************************************************************/

 534    4         4         PLANEFAILEVENT: PROCEDURE;
 535    5         5           FAILEDAC=ACFAILQPT(0);
 536    5         5           BASEFAILEDAC=BASE(ACFAILQPT(0));
 537    5         5           ACFAILQPT(0)=ACFAILQPT(ACFAILQPT(0));
```

```
STMT LEVEL NEST BLOCK PLVL  SOURCE TEXT

 538    5           5              ACFAILOBPT(ACFAILOFPT(0))=0;
 539    5           5              IF ACFAILOFPT(0)=0 THEN
 540    5           5                 DO:
 541    5     1     5                    ACFAILTIME=2**LIMTEST*HALT+1;
 542    5     1     5                    ACFAILOBOT=0;
 543    5     1     5                 END;
 544    5           5                 ELSE ACFAILTIME=FAILTIME(ACFAILOFPT(0));
 545    5           5              FAILDEGREE=3;
 546    5           5              U1=RAND(U1);
 547    5           5              DO WHILE (U1>CCDIST(CC(TYPE(FAILEDAC)),FAILDEGREE));
 548    5     1     5                   FAILDEGREE=FAILDEGREE-1;
 549    5     1     5              END;
 550    5           5              DEMANDS(BASEFAILEDAC)=DEMANDS(BASEFAILEDAC)+1;
 551    5           5              IF STOCKLEVEL(BASEFAILEDAC)¬=0
 552    5           5                  THEN DO;
 553    5     1     5                    DEMSAT(BASEFAILEDAC)=DEMSAT(BASEFAILEDAC)+1;
 554    5     1     5                     CALL DISTRIBUTIONS(BASEFAILEDAC,'STOCK');
 555    5     1     5                      STOCKLEVEL(BASEFAILEDAC)=STOCKLEVEL(BASEFAILEDAC)-1;
 556    5     1     5                      LIFENG(1,FAILEDAC)=RNG(MTBR(CC(TYPE(FAILEDAC))),
                                                            RVS(BASEFAILEDAC));
 557    5     1     5                      CALL FAILGEN(FAILEDAC,SHIPNUM(BASEFAILEDAC),BASEFAILEDAC);
 558    5     1     5                  END;
 559    5           5                  ELSE DO;
 560    5     1     5                     CALL DISTRIBUTIONS(BASEFAILEDAC,'DAC');
 561    5     1     5                      DACNUM(BASEFAILEDAC)=DACNUM(BASEFAILEDAC)+1;
 562    5     1     5                    DAC(BASEFAILEDAC)=DAC(BASEFAILEDAC)+1;
 563    5     1     5                      IF DACQTOP(BASEFAILEDAC)¬=0
 564    5     1     5                         THEN DO;
 565    5     2     5                            DACQPT(DACQBOT(BASEFAILEDAC))=FAILEDAC;
 566    5     2     5                            DACQBOT(BASEFAILEDAC)=FAILEDAC;
 567    5     2     5                            DACQPT(FAILEDAC)=0;
 568    5     2     5                         END;
 569    5     1     5                         ELSE DO;
 570    5     2     5                            DACQTOP(BASEFAILEDAC)=FAILEDAC;
 571    5     2     5                            DACQBOT(BASEFAILEDAC)=FAILEDAC;
 572    5     2     5                            DACQPT(FAILEDAC)=0;
 573    5     2     5                         END;
 574    5     1     5                         BASEBO(BOQMT)=BASEFAILEDAC;
 575    5     1     5                         CALL DISTRIBUTIONS(BASEFAILEDAC,'BO');
 576    5     1     5                         BACKORDERS(BASEFAILEDAC)=BACKORDERS(BASEFAILEDAC)+1;
 577    5     1     5                         CALL BACKORDQ(BASEFAILEDAC);
 578    5     1     5                  END;
 579    5           5              DESTINBASE=NEXTECH(BASEFAILEDAC,FAILDEGREE);
 580    5           5              DUMMYTIME=0;
 581    5           5              IF HD(DESTINBASE)=0 & FAILDEGREE=1 THEN FAILDEGREE=-1;
 583    5           5              IF HD(DESTINBASE)>ABS(FAILDEGREE) THEN
 584    5           5                 DO;
 585    5     1     5                    IF SHIPNUM(BASEFAILEDAC)>0 THEN
 586    5     1     5                       DO;
 587    5     2     5                         DUMMYTIME=SHIPTIME(SHIPNUM(BASEFAILEDAC))-CLOCK;
 588    5     2     5                         NUMSENTENG(SHIPNUM(BASEFAILEDAC))=NUMSENTENG(SHIPNUM(
                                                 BASEFAILEDAC))+1;
 589    5     2     5                       END;
 590    5     1     5                    ELSE DUMMYTIME=TRAVTIME(BASEFAILEDAC,DESTINBASE);
 591    5     1     5                    IF DESTINBASE¬=PORT(BASEFAILEDAC) THEN
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 592    5    1    5                    DO:
 593    5    2    5                       DEMANDS(DESTINBASE)=DEMANDS(DESTINBASE)+1;
 594    5    2    5                       IF STOCKLEVEL(DESTINBASE)¬=0 THEN
 595    5    2    5                          DO;
 596    5    3    5                             DEMSAT(DESTINBASE)=DEMSAT(DESTINBASE)+1;
 597    5    3    5                             CALL AVAILQ(BASEFAILEDAC,DESTINBASE,0,0);
 598    5    3    5                             CALL DISTRIBUTIONS(DESTINBASE,'STOCK');
 599    5    3    5                             STOCKLEVEL(DESTINBASE)=STOCKLEVEL(DESTINBASE)-1;
 600    5    3    5                          END;
 601    5    2    5                       ELSE
                                             DO;
 602    5    3    5                             BASEBO(BOCNT)=BASEFAILEDAC;
 603    5    3    5                             CALL DISTRIBUTIONS(DESTINBASE,'BO');
 604    5    3    5                             BACKORDERS(DESTINBASE)=BACKORDERS(DESTINBASE)+1;
 605    5    3    5                             CALL BACKORDQ(DESTINBASE);
 606    5    3    5                          END;
 607    5    2    5                    END;
 608    5    1    5                 BASEFAILEDAC=DESTINBASE;
 609    5    1    5                 DESTINBASE=NEXTECH(BASEFAILEDAC,FAILDEGREE);
 610    5    1    5              END;
 611    5         5           IF DESTINBASE¬=BASEFAILEDAC
 612    5         5               THEN DO;
 613    5    1    5                 IF DESTINBASE¬=PORT(BASEFAILEDAC) THEN
 614    5    1    5                 DEMANDS(DESTINBASE)=DEMANDS(DESTINBASE)+1;
 615    5    1    5                   IF SHIPNUM(BASEFAILEDAC)¬=0
 616    5    1    5                    THEN DO;
 617    5    2    5                       CALL AVAILQ(DESTINBASE,PORT(BASEFAILEDAC),
                                                SHIPNUM(BASEFAILEDAC),REPAIRTIME(FAILDEGREE));
 618    5    2    5                       NUMSENTENG(SHIPNUM(BASEFAILEDAC))=
                                           NUMSENTENG(SHIPNUM(BASEFAILEDAC))+1;
 619    5    2    5                    END;
 620    5    1    5                       ELSE CALL AVAILQ(DESTINBASE,BASEFAILEDAC,0,
                                                 REPAIRTIME(FAILDEGREE)+DUMMYTIME);
 621    5    1    5                   IF DESTINBASE¬=PORT(BASEFAILEDAC)
 622    5    1    5                       THEN DO;
 623    5    2    5                           IF STOCKLEVEL(DESTINBASE)¬=0
 624    5    2    5                               THEN DO;
 625    5    3    5                          DEMSAT(DESTINBASE)=DEMSAT(DESTINBASE)+1;
 626    5    3    5                                   IF SHIPNUM(BASEFAILEDAC)¬=0
 627    5    3    5                                       THEN CALL AVAILQ(PORT(BASEFAILEDAC),
                                                            DESTINBASE,0,0);
 628    5    3    5                                       ELSE CALL AVAILQ(BASEFAILEDAC,
                                                            DESTINBASE,0,0);
 629    5    3    5                                   CALL DISTRIBUTIONS(DESTINBASE,'STOCK');
 630    5    3    5                                   STOCKLEVEL(DESTINBASE) =
                                                   STOCKLEVEL(DESTINBASE)-1;
 631    5    3    5                               END;
 632    5    2    5                               ELSE DO;
 633    5    3    5                                   IF SHIPNUM(BASEFAILEDAC)¬=0
 634    5    3    5                                       THEN BASEBO(BOCNT)=PORT(BASEFAILEDAC);
 635    5    3    5                                       ELSE BASEBO(BOCNT)=BASEFAILEDAC;
 636    5    3    5                                   CALL DISTRIBUTIONS(DESTINBASE,'BO');
 637    5    3    5                                   BACKORDERS(DESTINBASE) =
                                                   BACKORDERS(DESTINBASE)+1;
 638    5    3    5                                   CALL BACKORDQ(DESTINBASE);
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 639    5    3     5                          END;
 640    5    2     5                       END;
 641    5    1     5                 END;
 642    5          5          ELSE CALL AVAILQ(BASEFAILEDAC,DESTINBASE,0,REPAIRTIME(FAILDEGREE));
 643    5          5          END PLANEFAILEVENT;


                              /*****************************************************************/
                              /*ENGAVAILEVENT:                                                  */
                              /*ENGAVAILEVENT IS CALLED WHEN AN ENGINE BECOMES AVAILABLE AT    */
                              /*A BASE.      THE AVAILABLE ENGINE IS REMOVED FROM THE AVAILA-  */
                              /*BILITY QUEUE.                                                   */
                              /*    1)   BACKORDERS ARE SATISFIED ON A FCFS BASIS.              */
                              /*             IF THE NEXT BACKORDER IS TO THE BASE ITSELF (A    */
                              /*             DOWNED AIRCRAFT), THE AIRCRAFT IS MADE OPERATION-*/
                              /*             AL AND PLACED BACK IN THE FAILURE QUEUE.           */
                              /*             IF THE NEXT BACKORDER IS TO ANOTHER BASE, THE     */
                              /*             ENGINE IS MADE AVAILABLE TO THAT BASE.             */
                              /*    2)   OTHERWISE THE ENGINE IS PLACED IN THE BASE'S OWN      */
                              /*         ENGINE STOCK IF NO BAKCORDERS EXIST.                   */
                              /*         DISTRIBUTIONS AND LEVELS ARE ADJUSTED ACCORDINGLY     */
                              /*         AFTER EACH TRANSACTION.                                */
                              /*****************************************************************/

 644    4          4         ENGAVAILEVENT: PROCEDURE;
 645    5          6           BASEAVAIL=BASENGAVAIL(ENGAVAILQFPT(0));
 646    5          6           ENGAVAILQHOLD=ENGAVAILQFPT(ENGAVAILQFPT(0));
 647    5          6           ENGAVAILQBPT(ENGAVAILQHOLD)=0;
 648    5          6           ENGAVAILQFPT(ENGAVAILQFPT(0))=ENGAVAILQMT;
 649    5          6           ENGAVAILQMT=ENGAVAILQFPT(0);
 650    5          6           ENGAVAILQFPT(0)=ENGAVAILQHOLD;
 651    5          6           IF ENGAVAILQFPT(0)=0 THEN ENGAVAILQBOT=0;
 653    5          6           IF BOQTOP(BASEAVAIL)¬=0
 654    5          6               THEN DO;
 655    5    1     6                   IF BASEBO(BOQTOP(BASEAVAIL))=BASEAVAIL
 656    5    1     6                       THEN DO;
 657    5    2     6                       IF DACQTOP(BASEAVAIL)¬=0
 658    5    2     6                           THEN DO;
 659    5    3     6                              IF CLOCK-LDACTIME(BASEAVAIL)>0
 660    5    3     6                                 THEN CALL DISTRIBUTIONS(BASEAVAIL,'DAC');
 661    5    3     6                              DACNUM(BASEAVAIL)=DACNUM(BASEAVAIL)-1;
 662    5    3     6                               LIFENG(1,DACQTOP(BASEAVAIL))=
                                                       RNG(MTBR(CC(TYPE(DACQTOP(BASEAVAIL)))),
                                                           RVS(BASEAVAIL));
 663    5    3     6                               CALL FAILGEN(DACQTOP(BASEAVAIL),
                                                       SHIPNUM(BASEAVAIL),BASEAVAIL);
 664    5    3     6                               DACQTOP(BASEAVAIL)=DACQPT(DACQTOP(BASEAVAIL));
 665    5    3     6                           END;
 666    5    2     6                           ELSE DO;
 667    5    3     6                             IF CLOCK-LSTIME(BASEAVAIL)>0
 668    5    3     6                                THEN CALL DISTRIBUTIONS(BASEAVAIL,'STOCK');
 669    5    3     6                               STOCKLEVEL(BASEAVAIL)=STOCKLEVEL(BASEAVAIL)+1;
 670    5    3     6                           END;
 671    5    2     6                       END;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 672    5    1    6               ELSE CALL AVAILQ(BASEPQ(BOQTOP(BASEAVAIL)),BASEAVAIL,0,0);
 673    5    1    6               BOQHOLD=BOQPT(BOQTOP(BASEAVAIL));
 674    5    1    6               BOQPT(BOQTOP(BASEAVAIL))=BOQMT;
 675    5    1    6               BOQMT=BOQTOP(BASEAVAIL);
 676    5    1    6               BOQTOP(BASEAVAIL)=BOQHOLD;
 677    5    1    6          IF CLOCK-LBOTIME(BASEAVAIL)>0 THEN CALL
                             DISTRIBUTIONS(BASEAVAIL,'BO');
 679    5    1    6          BACKORDERS(BASEAVAIL)=BACKORDERS(BASEAVAIL)-1;
 680    5    1    6          END;
 681    5         6          ELSE DO:
 682    5    1    6             IF CLOCK-LSTIME(BASEAVAIL)>0 THEN CALL
                             DISTRIBUTIONS(BASEAVAIL,'STOCK');
 684    5    1    6                  STOCKLEVEL(BASEAVAIL)=STOCKLEVEL(BASEAVAIL)+1;
 685    5    1    6               END;
 686    5         6          IF ENGAVAILQFPT(0)=0 THEN NJUNAVAILTIME=2**LIMTEST*HALT+1;
 688    5         6          ELSE NJUNAVAILTIME=ENGAVAILTIME(ENGAVAILQFPT(0));
 689    5         6          END ENGAVAILEVENT;


                             /*********************************************************************/
                             /*SHIPEVENT:                                                         */
                             /*A SHIPTIME WILL INDICATE A SHIP'S TIME OF NEXT DOCKING OR          */
                             /*DEPLOYMENT, DEPENDING ON ITS STATE AT SHIPTIME.                    */
                             /*IF THE SHIP WAS 'AT SEA', SHIPTIME INDICATES A DOCKING TIME.       */
                             /*   AT THIS TIME THE SHIP'S STATISTICS, LEVELS AND DISTRIBUTION*/
                             /*   ARE UPDATED TO ACCOUNT FOR ITS 'DORMANCY' DURING ITS TIME       */
                             /*   IN PORT.                                                        */
                             /*   (IN THE REAL SYSTEM, AT THIS POINT, ALL THE FAILED ENGINES      */
                             /*   ARE SENT OFF TO BE REPAIRED.  THE SIMULATION HAS ALREADY        */
                             /*   TAKEN CARE OF THIS AT THE TIME OF FAILURE).                     */
                             /*   THE NEXT DOCKTIME IS REMOVED FROM THE SHIP SCHEDULE AND A       */
                             /*   NEW ONE IS GENERATED.  THE NEW SHIPTIME IS CALCULATED AND       */
                             /*   THE SHIP STATE BECOMES 'AT PORT'.                               */
                             /*IF THE SHIP IS 'AT PORT' AT SHIP TIME, IT MUST BE PREPARED TO      */
                             /*DEPLOY.                                                            */
                             /*   ITS NEW SHIPTIME IS DETERMINED, AND ITS STATE IS CHANGED TO*/
                             /*   'AT SEA'.  THE SHIP SCHEDULE ENTRY IS LIKEWISE REGENERATED.*/
                             /*   THE SHIP MUST BE RESTOCKED BY THE PORT.                         */
                             /*   THE ORDER OF RESTOCKING IS                                      */
                             /*   1)   REJUVENATE DOWNED AIRCRAFT ON BOARD.                       */
                             /*        DETERMINE THE NEW FAILURE TIME FOR THE AIRCRAFT.           */
                             /*        PLACE IT BACK IN THE FAILURE QUEUE.                        */
                             /*   2)   REPLACE THE ENGINE STOCK TO THE SHIP'S CAPACITY.           */
                             /*        IF, AT ANY TIME DURING THE RESTOCKING, THE PORT RUNS       */
                             /*        OUT OF ENGINES, THE APPROPRIATE NUMBER OF BACKORDERS       */
                             /*        ARE LOGGED AT THE PORT, AND REFILLED THE NEXT TIME.        */
                             /*        LEVELS AND DISTRIBUTIONS ARE MAINTAINED THROUGHOUT THE*/
                             /*        STOCKING PROCESS.                                          */
                             /*********************************************************************/

 690    4         4        SHIPEVENT: PROCEDURE;
 691    5         7                  BASENUM=SHIPBASE(SHIPQPT(0));
 692    5         7          IF STATE(SHIPQPT(0))='AT SEA'
 693    5         7             THEN DO;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 694    5    1     7            CUMSEATIME(SHIPQPT(0))=CUMSEATIME(SHIPQPT(0))+CLOCK-
                                                  LSEATIME(SHIPQPT(0));
 695    5    1     7            SAMPSEATIME(SHIPQPT(0),NUMSAMPS)=SAMPSEATIME(SHIPQPT(0),
                                NUMSAMPS)+CLOCK-LSEATIME(SHIPQPT(0));
 696    5    1     7            STATE(SHIPQPT(0))='AT PORT';
 697    5    1     7            SHIPTIME(SHIPQPT(0))=SHIPTIME(SHIPQPT(0))
                                +PORTSCHED(SHIPQPT(0),IND(SHIPQPT(0)));
 698    5    1     7           IF SCHED_IND(SHIPQPT(0))=0 THEN
 699    5    1     7            PORTSCHED(SHIPQPT(0),IND(SHIPQPT(0)))=
                          SHIPTIME_RNG(MPORTIME(SHIPQPT(0)),STD_DEV(SHIPQPT(0)),
                          RVS(BASENUM));
 700    5    1     7           IND(SHIPQPT(0))=MOD(IND(SHIPQPT(0)),SCHED_LENGTH(SHIPQPT(0)))
                               +1;
 701    5    1     7          CALL DISTRIBUTIONS(BASENUM,'STOCK');
 702    5    1     7          CALL DISTRIBUTIONS(BASENUM,'BO');
 703    5    1     7          CALL DISTRIBUTIONS(BASENUM,'DAC');
 704    5    1     7           LSTIME(BASENUM)=SHIPTIME(SHIPQPT(0));
 705    5    1     7           LBOTIME(BASENUM)=SHIPTIME(SHIPQPT(0));
 706    5    1     7           LDACTIME(BASENUM)=SHIPTIME(SHIPQPT(0));
 707    5    1     7        END;
 708    5          7        ELSE DO:
 709    5    1     7            LSEATIME(SHIPQPT(0))=CLOCK;
 710    5    1     7            STATE(SHIPQPT(0))='AT SEA';
 711    5    1     7            SHIPTIME(SHIPQPT(0))=SHIPTIME(SHIPQPT(0)) +
                                SEASCHED(SHIPQPT(0),IND(SHIPQPT(0)));
 712    5    1     7           IF SCHED_IND(SHIPQPT(0))=0 THEN
 713    5    1     7            SEASCHED(SHIPQPT(0),IND(SHIPQPT(0)))=
                          SHIPTIME_RNG(MSEATIME(SHIPQPT(0)),STD_DEV(SHIPQPT(0)),
                          RVS(BASENUM));
 714    5    1     7          TEMP=NUMSENTENG(SHIPQPT(0));
 715    5    1     7          DEMANDS(PORT(BASENUM))=DEMANDS(PORT(BASENUM))+
                          NUMSENTENG(SHIPQPT(0))-LNUMSENTENG(SHIPQPT(0));
 716    5    1     7          CALL DISTRIBUTIONS(PORT(BASENUM),'STOCK');
 717    5    1     7          DO WHILE(STOCKLEVEL(PORT(BASENUM))¬=0 &
                          NUMSENTENG(SHIPQPT(0))>0);
 718    5    2     7              IF DACQTOP(BASENUM)¬=0
 719    5    2     7                 THEN DO;
 720    5    3     7                LIFENG(1,DACQTOP(BASENUM))=
                                RNG(MTBR(CC(TYPE(DACQTOP(BASENUM)))),RVS(BASENUM));
 721    5    3     7                CALL FAILGEN(DACQTOP(BASENUM),SHIPQPT(0),BASENUM);
 722    5    3     7                DACQTOP(BASENUM)=DACQPT(DACQTOP(BASENUM));
 723    5    3     7                DACNUM(BASENUM)=DACNUM(BASENUM)-1;
 724    5    3     7                BACKORDERS(BASENUM)=BACKORDERS(BASENUM)-1;
 725    5    3     7                BOQHOLD=BOQPT(BOQTOP(BASENUM));
 726    5    3     7                BOQPT(BOQTOP(BASENUM))=BOQMT;
 727    5    3     7                BOQMT=BOQTOP(BASENUM);
 728    5    3     7                BOQTOP(BASENUM)=BOQHOLD;
 729    5    3     7            END;
 730    5    2     7                 ELSE DO;
 731    5    3     7                STOCKLEVEL(BASENUM)=STOCKLEVEL(BASENUM)+1;
 732    5    3     7                 END;
 733    5    2     7                STOCKLEVEL(PORT(BASENUM))=STOCKLEVEL(PORT(BASENUM))-1;
 734    5    2     7                    NUMSENTENG(SHIPQPT(0))=NUMSENTENG(SHIPQPT(0))-1;
 735    5    2     7                 END;
 736    5    1     7            IF TEMP-NUMSENTENG(SHIPQPT(0))>LNUMSENTENG(SHIPQPT(0))
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 737    5    1    7                  THEN DEMSAT(PORT(BASENUM))=DEMSAT(PORT(BASENUM))
                                     +TEMP-NUMSENTENG(SHIPQPT(0))-LNUMSENTENG(SHIPQPT(0));
 738    5    1    7                  CALL DISTRIBUTIONS(PORT(BASENUM),'SO');
 739    5    1    7                  BACKORDERS(PORT(BASENUM))=BACKORDERS(PORT(BASENUM))+
                                     NUMSENTENG(SHIPQPT(0))-LNUMSENTENG(SHIPQPT(0));
 740    5    1    7                IF NUMSENTENG(SHIPQPT(0))>LNUMSENTENG(SHIPQPT(0))
 741    5    1    7                 THEN BO(PORT(BASENUM))=BO(PORT(BASENUM))+NUMSENTENG
                                    (SHIPQPT(0))-LNUMSENTENG(SHIPQPT(0));
 742    5    1    7                  LNUMSENTENG(SHIPQPT(0))=NUMSENTENG(SHIPQPT(0));
 743    5    1    7              END;
 744    5         7              IF SHIPTIME(SHIPQPT(0))>SHIPTIME(SHIPQPT(SHIPQPT(0)))
 745    5         7                  THEN DO;
 746    5    1    7                      HOLD1=SHIPQPT(0);
 747    5    1    7                      SHIPQPT(0)=SHIPQPT(SHIPQPT(0));
 748    5    1    7                  LINSRT=SHIPQPT(0);
 749    5    1    7                      INSRT=SHIPQPT(LINSRT);
 750    5    1    7                      DO WHILE (SHIPTIME(INSRT)<SHIPTIME(HOLD1));
 751    5    2    7                          LINSRT=INSRT;
 752    5    2    7                          INSRT=SHIPQPT(INSRT);
 753    5    2    7                      END;
 754    5    1    7                      SHIPQPT(LINSRT)=HOLD1;
 755    5    1    7                      SHIPQPT(HOLD1)=INSRT;
 756    5    1    7                  END;
 757    5         7          END SHIPEVENT;


                             /***************************************************************/
                             /*BACKORDQ:                                                    */
                             /*THIS PROCEDURE QUEUES BACKORDERS TO A SPECIFIC BASE.  THE    */
                             /*QUEUE AT A BASE IS MAINTAINED SO THAT FIRST BACKORDERS ARE   */
                             /*SATISFIED FIRST.                                             */
                             /***************************************************************/

 758    4         4         BACKORDQ: PROCEDURE(BASE);
 759    5         8           DCL BASE BIN FLOAT;
 760    5         8           BO(BASE)=BO(BASE)+1;
 761    5         8           BOQHOLD=BOQPT(BOQMT);
 762    5         8           IF BOQTOP(BASE)¬=0
 763    5         8              THEN DO;
 764    5    1    8                  BOQPT(BOQBOT(BASE))=BOQMT;
 765    5    1    8                  BOQBOT(BASE)=BOQMT;
 766    5    1    8           BOQPT(BOQMT)=0;
 767    5    1    8              END;
 768    5         8              ELSE DO;
 769    5    1    8                  BOQTOP(BASE)=BOQMT;
 770    5    1    8                  BOQBOT(BASE)=BOQMT;
 771    5    1    8                  BOQPT(BOQMT)=0;
 772    5    1    8              END;
 773    5         8           BOQMT=BOQHOLD;
 774    5         8           END BACKORDQ;


                             /***************************************************************/
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                               /*FAILGEN:                                                          */
                               /*FAILGEN IS CALLED WHEN AN AIRCRAFT IS PUT BACK IN OPERATION.      */
                               /*IT IDENTIFIES THE ENGINE WHICH WILL CAUSE THE AIRCRAFT TO FAIL*/
                               /*GIVEN A FLYING SCHEDULE.  THE OTHER ENGINE(S)' LIVES ARE RE-      */
                               /*DUCED BY THE LIFE OF THE ENGINE WHICH WILL FAIL FIRST.            */
                               /*THE NUMBER OF DAYS BEFORE FAILURE DEPENDS ON THE TYPE OF BASE */
                               /*AT WHICH AN AIRCRAFT IS LOCATED.                                  */
                               /*IF AN AIRCRAFT FLIES ON A CARRIER, THE DAYS UNTIL FAILURE MUST*/
                               /*BE 'PADDED' BY A PORT TIME, WHEN THE CARRIER SUPPORTS NO          */
                               /*FLYING.  A GROUND BASE SUPPORTS FLYING MISSIONS AT ALL            */
                               /*TIMES.                                                            */
                               /*IT IS ASSUMED THAT AN AIRCRAFT FLIES THE SAME NUMBER OF HOURS */
                               /*EACH DAY.                                                         */
                               /*AFTER A FAILURE TIME IS DETERMINED, THE AIRCRAFT IS PLACED IN */
                               /*THE FAILURE QUEUE.                                                */
                               /******************************************************************/

 775    4          4          FAILGEN:PROCEDURE(PLANEN,SHIPN,BASEN);
 776    5          9            DCL(PLANEN,SHIPN,BASEN) BIN FLOAT;
 777    5          9            MDFH=FOHM(TYPE(PLANEN),BASEN)/30;
 778    5          9            CUMFLYTIME=0;
 779    5          9            FLYHRS=10*2**LIMTEST*HALT+1;
 780    5          9            DAYSTILFAIL=0; FLYTIME=0; SWITCH=1;
 783    5          9            DO NUMENG=1 TO ENGCAP(TYPE(PLANEN));
 784    5     1    9                IF FLYHRS>LIFENG(NUMENG,PLANEN)
 785    5     1    9                    THEN DO;
 786    5     2    9                        FLYHRS=LIFENG(NUMENG,PLANEN);
 787    5     2    9                        NUMSAVE=NUMENG;
 788    5     2    9                    END;
 789    5     1    9            END;
 790    5          9            DO NUMENG=1 TO ENGCAP(TYPE(PLANEN));
 791    5     1    9                LIFENG(NUMENG,PLANEN)=LIFENG(NUMENG,PLANEN)-FLYHRS;
 792    5     1    9            END;
 793    5          9            IF NUMSAVE¬=1
 794    5          9                THEN DO;
 795    5     1    9                    HOLD=LIFENG(1,PLANEN);
 796    5     1    9                    LIFENG(1,PLANEN)=LIFENG(NUMSAVE,PLANEN);
 797    5     1    9                    LIFENG(NUMSAVE,PLANEN)=HOLD;
 798    5     1    9                END;
 799    5          9            DAYSTILFAIL=FLYHRS*30/FOHM(TYPE(PLANEN),BASEN);
 800    5          9            HOLD=CLOCK;
 801    5          9            IF SHIPN¬=0 THEN
 802    5          9               DO;
 803    5     1    9                  IF STATE(SHIPN)='AT PORT' THEN
 804    5     1    9                     DO;
 805    5     2    9                        DUM=IND(SHIPN);
 806    5     2    9                        HOLD=SHIPTIME(SHIPN);
 807    5     2    9                        TIMLIM=SEASCHED(SHIPN,DUM);
 808    5     2    9                     END;
 809    5     1    9                  ELSE
                                         DO;
 810    5     2    9                        DUM=IND(SHIPN);
 811    5     2    9                        TIMLIM=SHIPTIME(SHIPN)-CLOCK;
 812    5     2    9                     END;
 813    5     1    9                     DO WHILE(DAYSTILFAIL>TIMLIM);
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 814    5    2    9                      CUMFLYTIME=TIMLIM+CUMFLYTIME+PORTSCHED(SHIPN,DUM);
 815    5    2    9                       DUM=MOD(DUM,SCHED_LENGTH(SHIPN))+1;
 816    5    2    9                       DAYSTILFAIL=DAYSTILFAIL-TIMLIM;
 817    5    2    9                       TIMLIM=SEASCHED(SHIPN,DUM);
 818    5    2    9                    END;
 819    5    1    9              END;
 820    5         9          FAILTIME(PLANEN)=HOLD+CUMFLYTIME+DAYSTILFAIL;
 821    5         9          ACFAILQDUM=ACFAILQBOT;
 822    5         9          DO WHILE(FAILTIME(PLANEN)<FAILTIME(ACFAILQDUM));
 823    5    1    9               ACFAILQDUM=ACFAILQBPT(ACFAILQDUM);
 824    5    1    9          END;
 825    5         9          IF ACFAILQDUM=ACFAILQBOT
 826    5         9               THEN DO;
 827    5    1    9                     ACFAILQFPT(ACFAILQBOT)=PLANEN;
 828    5    1    9                     ACFAILQFPT(PLANEN)=PLANEN;
 829    5    1    9                     ACFAILQBPT(PLANEN)=ACFAILQBOT;
 830    5    1    9                     ACFAILQBOT=PLANEN;
 831    5    1    9               END;
 832    5         9               ELSE DO;
 833    5    1    9                     ACFAILQFPT(PLANEN)=ACFAILQFPT(ACFAILQDUM);
 834    5    1    9                     ACFAILQFPT(ACFAILQDUM)=PLANEN;
 835    5    1    9                     ACFAILQBPT(PLANEN)=ACFAILQDUM;
 836    5    1    9                     ACFAILQBPT(ACFAILQFPT(PLANEN))=PLANEN;
 837    5    1    9               END;
 838    5         9          ACFAILTIME=FAILTIME(ACFAILQFPT(0));
 839    5         9          END FAILGEN;


                             /****************************************************************/
                             /* SAMPLE:                                                      */
                             /* SAMPLE GATHERS SEPARATE DATA OBSERVATIONS AS DETERMINED BY   */
                             /*THE CURRENT SAMPLE INTERVAL                                   */
                             /****************************************************************/

 840    4         4      SAMPLE:PROCEDURE;
 841    5        10          IF CLOCK=RUNIN THEN
 842    5        10              DO;
 843    5    1   10                 CUMSEATIME=0;
 844    5    1   10                 SAMPSEATIME=0;
 845    5    1   10                 DEMSAT=0;
 846    5    1   10                 DEMANDS=0;
 847    5    1   10                 BO=BACKORDERS;
 848    5    1   10                 DAC=DACNUM;
 849    5    1   10                 LDACTIME=CLOCK;
 850    5    1   10                 LBOTIME=CLOCK;
 851    5    1   10                 LSTIME=CLOCK;
 852    5    1   10                 LSTBCINDTIME=CLOCK;
 853    5    1   10                 LSEATIME=CLOCK;
 854    5    1   10              END;
 855    5        10          ELSE
                                DO;
 856    5    1   10                 DO I=1 TO TOTNUMBASES;
 857    5    2   10                 IF LDACTIME(I)-CLOCK<0 THEN CALL DISTRIBUTIONS(I,'DAC');
 858    5    2   10                 IF SHIPNUM(I)>0 THEN IF STATE(SHIPNUM(I))='AT SEA' THEN
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 861    5    2    10                  DO:
 862    5    3    10                     CUMSEATIME(SHIPNUM(I))=CUMSEATIME(SHIPNUM(I))+CLOCK-
                                         LSEATIME(SHIPNUM(I));
 863    5    3    10                     SAMPSEATIME(SHIPNUM(I),NUMSAMPS)=SAMPSEATIME(SHIPNUM(I),
                                         NUMSAMPS)+CLOCK-LSEATIME(SHIPNUM(I));
 864    5    3    10                     LSEATIME(SHIPNUM(I))=CLOCK;
 865    5    3    10                  END;
 866    5    2    10               END;
 867    5    1    10                IF CLOCK=HALT THEN DO;
 869    5    2    10                   PASSBASE=0;
 870    5    2    10                   DO K=1 TO TOTNUMBASES;
 871    5    3    10                   IF AIRCRAFT(K)>0 THEN
 872    5    3    10                      PASSFAIL(K)=IND_TEST(DACAREA(K,*),K);
 873    5    3    10                   END;
 874    5    2    10                   IF CRIT_BASE>0 THEN DO;
 876    5    3    10                      PF=PASSFAIL(CRIT_BASE);
 877    5    3    10                      CALL IND_OUT;
 878    5    3    10                   END;
 879    5    2    10                   ELSE DO;
 880    5    3    10                      DO K=1 TO TOTNUMBASES;
 881    5    4    10                         PASSBASE=PASSBASE+PASSFAIL(K);
 882    5    4    10                      END;
 883    5    3    10                      PF=PASSBASE/TOTNUMACB;
 884    5    3    10                      CALL IND_OUT;
 885    5    3    10                   END;
 886    5    2    10                END;
 887    5    1    10               NUMSAMPS=NUMSAMPS+1;
 888    5    1    10               DACAREA(*,NUMSAMPS)=0;
 889    5    1    10               SAMPSEATIME(*,NUMSAMPS)=0;
 890    5    1    10            END;
 891    5         10         SAMPTIME=SAMPTIME+SI;
 892    5         10      END SAMPLE;

                             /*****************************************************************/
                             /* IND_TEST:                                                     */
                             /* PERFORMS THE TEST OF INDEPENDENCE ON THE TOTAL NUMBER OF SAM-*/
                             /*PLES WHICH WERE GATHERED IN THE SIMULATION                     */
                             /*****************************************************************/

 893    4          4       IND_TEST: PROCEDURE(DACAREA,CRIT_BASE);
 894    5         11         DCL (DACAREA(*),CRIT_BASE) BIN FLOAT;
 895    5         11         DCL(XBAR,I,SUM1,SUM2) INIT(0) BIN FLOAT;
 896    5         11            DO I=1 TO TOTSAMPLES;
 897    5    1    11               IF SHIPNUM(CRIT_BASE)>0 THEN SA=SAMPSEATIME(SHIPNUM(CRIT_BASE)
                                   ,I);
 899    5    1    11               ELSE SA=SI;
 900    5    1    11         XBAR=XBAR+DACAREA(I)/SA;
 901    5    1    11            END;
 902    5         11         XBAR=XBAR/TOTSAMPLES;
 903    5         11            DO I=1 TO TOTSAMPLES-1;
 904    5    1    11               IF SHIPNUM(CRIT_BASE)>0 THEN
 905    5    1    11               DO:
 906    5    2    11               SUM1=SUM1+(DACAREA(I)/SAMPSEATIME(SHIPNUM(CRIT_BASE)
                                   ,I)-DACAREA(I+1)/SAMPSEATIME(SHIPNUM(CRIT_BASE),I+1)
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                       )**2;
 907    5    2     11                  SUM2=SUM2+(DACAREA(I)/SAMPSEATIME(SHIPNUM(CRIT_BASE)
                                       ,I)-XBAR)**2;
 908    5    2     11                  END;
 909    5    1     11                  ELSE DO:
 910    5    2     11                  SUM1=SUM1+(DACAREA(I)/SI-DACAREA(I+1)/SI)**2;
 911    5    2     11                  SUM2=SUM2+(DACAREA(I)/SI-XBAR)**2;
 912    5    2     11                  END;
 913    5    1     11               END;
 914    5          11                  IF SHIPNUM(CRIT_BASE)>0 THEN SA=SAMPSEATIME(SHIPNUM(CRIT_BASE)
                                       ,TOTSAMPLES);
 916    5          11                  ELSE SA=SI;
 917    5          11                  SUM2=SUM2+(DACAREA(TOTSAMPLES)/SA-XBAR)**2;
 918    5          11            IF SUM2>0 THEN
 919    5          11            CN(CRIT_BASE)=1-(SUM1/(2*SUM2));
 920    5          11            ELSE CN(CRIT_BASE)=0;
 921    5          11            IF CN(CRIT_BASE)<=NORMAL_STAT*SQRT((TOTSAMPLES-2)/
                                 (TOTSAMPLES**2-1)) THEN
 922    5          11               RETURN(1); ELSE RETURN(0);
 924    5          11            END IND_TEST;


                                 /**********************************************************************/
                                 /* IND_OUT:                                                           */
                                 /* INDOUT PRINTS THE  SYSTEM RESULTS FOR THE INDEPENDENCE TESTS       */
                                 /**********************************************************************/

 925    4           4          IND_OUT: PROCEDURE;
 926    5          12            IF PF>=PERCPASS
 927    5          12               THEN DO;
 928    5    1     12            PUT PAGE EDIT ('INDEPENDENCE TEST',NUMFAILS+1,PF*100,
                                 '% OF REQUIRED BASES HAVE PASSED INDEPENDENCE TESTS WITH A',
                                 ALPHANORM,'SIGNIFICANCE LEVEL.',TOTSAMPLES,
                                 'SAMPLES WERE OBSERVED.',  SI,
                                 'DAYS COMPRISED INDEPENDENT SAMPLE INTERVAL.') (COL(20),A,X(1),F(2),
                                 COL(15),F(6,2),X(1),A,X(1),F(4,2),X(1),A,COL(15),F(5),X(1),A,
                                 COL(15),F(5),X(1),A);
 929    5    1     12            CALL RES_OUT;
 930    5    1     12            CALL REPORT;
 931    5    1     12            END;
 932    5          12            ELSE DO;
 933    5    1     12            PUT PAGE EDIT ('INDEPENDENCE TEST',NUMFAILS+1,PF*100,
                                 '% OF REQUIRED BASES HAVE PASSED INDEPENDENCE TESTS WITH A ',
                                 ALPHANORM,'SIGNIFICANCE LEVEL.',
                                 'THIS DOES NOT MEET THE USER''S',PERCPASS*100,'% REQUIRED.',
                                 TOTSAMPLES, 'SAMPLES WERE OBSERVED.',SI,
                                 'DAYS COMPRISED THE SAMPLE INTERVAL.')
                                 (COL(20),A,X(1),F(2),COL(15),F(6,2),A,F(4,2),X(1),A,COL(15),A,X(1),
                                 F(6,2),X(1),A,COL(15),F(5),X(1),A,COL(15),F(5),X(1),A);
 934    5    1     12            CALL RES_OUT;
 935    5    1     12                  NUMFAILS=NUMFAILS+1;
 936    5    1     12                  IF NUMFAILS<LIMTEST THEN
 937    5    1     12                     DO;
 938    5    2     12                        DO I=1 TO TOTSAMPLES/2;
 939    5    3     12                           DACAREA(*,I)=DACAREA(*,2*I-1)+DACAREA(*,2*I);
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

 940    5     3    12                       SAMPSEATIME(*,I)=SAMPSEATIME(*,2*I-1)+SAMPSEATIME(
                                            *,2*I);
 941    5     3    12                    END;
 942    5     2    12                    NUMSAMPS=I-1;
 943    5     2    12                    SI=2*SI;
 944    5     2    12                    HALT=HALT+SI*.5*TOTSAMPLES;
 945    5     2    12                 END;
 946    5     1    12              ELSE
                                      DO:
 947    5     2    12         PUT SKIP (5) EDIT ('THE LIMIT FOR INDEPENDENCE TESTS HAS BEEN ',
                              'REACHED.  NO INDEPENDENT OBSERVATIONS WERE MADE WITHIN PRE-',
                              'SCRIBED PARAMETERS.  SIMULATION STATISTICS WHICH FOLLOW',
                              'WILL BE BIASED.', 'POSSIBLE REMEDIES INCLUDE:',
                              '1)  INCREASE LIMITS FOR NUMBER OF TESTS.',
                              '2)  RELAX SIGNIFICANCE LEVEL ON INDEPENDENCE TEST.  (SEE '||
                              'SPAERS REPORT FOR IMPLICATIONS).')
                              (COL(20),A,COL(15),A,COL(15),A,COL(15),A,COL(20),A,COL(25),A,
                              COL(25),A);
 948    5     2    12         CALL REPORT;


 949    5     2    12         RES_OUT: PROCEDURE;
 950    6     2    13         PUT SKIP(5) EDIT ('RESULTS OF INDEPENDENCE TESTS AT INDIVIDUAL '||
                              'BASES.',
                              'OUTCOME OF ','BASE','INDEPENDENCE TESTS','TEST STATISTIC (CN)')
                              (COL(20),A,SKIP(2),COL(39),A,COL(19),A,COL(35),A,COL(65),A,SKIP(2));
 951    6     2    13         DO JK=1 TO TOTNUMBASES;
 952    6     3    13         IF (CRIT_BASE=0|CRIT_BASE=JK) & AIRCRAFT(JK)>0
 953    6     3    13            THEN PUT SKIP EDIT ('*') (COL(15),A);
 954    6     3    13            PUT EDIT (BASENAME(JK)) (COL(16),A);
 955    6     3    13            IF AIRCRAFT(JK)>0
 956    6     3    13               THEN DO;
 957    6     4    13                  IF PASSFAIL(JK)=1
 958    6     4    13                     THEN PUT EDIT ('PASS') (COL(43),A);
 959    6     4    13                     ELSE PUT EDIT ('FAIL') (COL(43),A);
 960    6     4    13               END;
 961    6     3    13               ELSE PUT EDIT ('NO FLYING ACTIVITY') (COL(35),A);
 962    6     3    13               IF AIRCRAFT(JK)>0
 963    6     3    13                  THEN PUT EDIT (CN(JK))       (COL(69),F(6,3));
 964    6     3    13                  ELSE PUT EDIT ('------')(COL(69),A);
 965    6     3    13         END;
 966    6     2    13         PUT SKIP(2) EDIT ('*',PERCPASS*100,'% OF BASES INDICATED MUST '||
                              'PASS INDEPENDENCE TESTS.') (COL(15),A,X(1),F(6,2),A);
 967    6     2    13         PUT EDIT('INDEPENDENCE TEST PASS:  CN<=',NORMAL_STAT*FACS)
                              (COL(15),A,F(6,3));
 968    6     2    13         END RES_OUT;
 969    5     2    12         END IND_OUT;

                              /*****************************************************************/
                              /*REPORT:                                                        */
                              /*REPORT PRINTS RELEVANT AVERAGES, RATIOS AND DISTRIBUTIONS AT   */
                              /*TIMES WHICH ARE MULTIPLES OF THE RUNIN TIME.  THE NUMBER OF    */
                              /*REPORTS AND THE MULTIPLE OF RUNIN TIME ARE INPUT VARIABLES     */
                              /*AVAILABLE TO THE USER.                                         */
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                  /********************************************************************/

 970    4          4          REPORT: PROCEDURE;
 971    5         14            DCL(II,DUMMY,XX,YY,AA,BB,BBB,WW,CC,BDB,NN,MM,PP,DUM1,DUM2,DUM3)
                                BIN FLOAT;
 972    5         14            DCL CHR CHAR(1) VAR;
 973    5         14            DCL(READYRATE,FILLRATE,ZERO,INDD) INIT(0) BIN FLOAT;
 974    5         14              DO I=1 TO TOTNUMBASES;
 975    5    1    14                    IF LSTIME(I)-CLOCK<0
 976    5    1    14                       THEN CALL DISTRIBUTIONS(I,'STOCK');
 977    5    1    14                    IF LBOTIME(I)-CLOCK<0
 978    5    1    14                       THEN CALL DISTRIBUTIONS(I,'BO');
 979    5    1    14              END;
 980    5         14            PU  PAGE;
 981    5         14                PUT EDIT('*****(THERE WAS A RUN IN PERIOD OF',RUNIN,
                                    ' DAYS)*****') (COL(40),A,F(4),A);
 982    5         14                PUT SKIP(2);
 983    5         14            DO II=1 TO 120;
 984    5    1    14              PUT EDIT('*') (A);
 985    5    1    14            END;
 986    5         14            PUT SKIP EDIT('FINAL STATISTICS') (COL(53),A);
 987    5         14            PUT SKIP EDIT ('SIMULATION TIME= ',CLOCK-RUNIN)(COL(49),A,F(8,2));
 988    5         14            PUT SKIP(3);
 989    5         14            DO I=1 TO TOTNUMBASES;
 990    5    1    14              IF AIRCRAFT(I)>0 THEN DO;
 992    5    2    14              DO J=1 TO TOTSAMPLES;
 993    5    3    14                IF SHIPNUM(I)>0 THEN SA=SAMPSEATIME(SHIPNUM(I),J);
 995    5    3    14                ELSE SA=SI;
 996    5    3    14                AVGDAC(I)=AVGDAC(I)+DACAREA(I,J)/SA;
 997    5    3    14              END;
 998    5    2    14              AVGDAC(I)=AVGDAC(I)/TOTSAMPLES;
 999    5    2    14              DO J=1 TO TOTSAMPLES;
1000    5    3    14               IF SHIPNUM(I)>0 THEN SA=SAMPSEATIME(SHIPNUM(I),J);
1002    5    3    14               ELSE SA=SI;
1003    5    3    14               VARCRITERION(I)=VARCRITERION(I)+(DACAREA(I,J)/(SA*AIRCRAFT(I))
                               -AVGDAC(I)/AIRCRAFT(I))**2;
1004    5    3    14             END;
1005    5    2    14             VARCRITERION(I)=VARCRITERION(I)/(TOTSAMPLES-1);
1006    5    2    14              END;
1007    5    1    14                DO II=1 TO 120;
1008    5    2    14                   PUT EDIT('-') (A);
1009    5    2    14                END;
1010    5    1    14            IF SHIPNUM(I)=0 THEN STIME=CLOCK-RUNIN;
1012    5    1    14            ELSE STIME=CUMSEATIME(SHIPNUM(I));
1013    5    1    14                ZERO=STOCKDIST(I,0)/STIME;
1014    5    1    14                IF STOCKDIST(I,0)>0 THEN
1015    5    1    14                         DO II=1 TO MAXSTOCK+1;
1016    5    2    14                            IF BODIST(I,II)>0 THEN
1017    5    2    14                            ZERO=ZERO-BODIST(I,II)/STIME;
1018    5    2    14                         END;
1019    5    1    14                READYRATE=ZERO;
1020    5    1    14                FILLRATE=0;
1021    5    1    14                   DO II=1 TO MAXSTOCK+1;
1022    5    2    14                      IF STOCKDIST(I,II)>0 THEN
1023    5    2    14                         DO;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1024    5    3    14                      READYRATE=READYRATE+STOCKDIST(I,II)/STIME;
1025    5    3    14                      FILLRATE=FILLRATE+STOCKDIST(I,II)/STIME;
1026    5    3    14                   END;
1027    5    2    14                END;
1028    5    1    14             PUT SKIP(2) EDIT('BASE# ',I,':     ',BASENAME(I))
                                 (COL(48),A,COL(54),F(2),A,A);
1029    5    1    14             PUT SKIP(1);
1030    5    1    14             SWITCH=0;
1031    5    1    14             DO J=1 TO TOTNUMBASES WHILE(SWITCH=0);
1032    5    2    14                IF PORT(J)=I THEN SWITCH=1;
1034    5    2    14             END;
1035    5    1    14             PUT SKIP EDIT('I.READY RATE= ') (COL(46),A);
1036    5    1    14             IF SWITCH=1 THEN PUT EDIT('*****') (A);
1038    5    1    14             ELSE PUT EDIT(READYRATE) (F(5,3));
1039    5    1    14             PUT SKIP EDIT('II.FILL RATE= ') (COL(45),A);
1040    5    1    14             IF SWITCH=1 THEN PUT EDIT('*****') (A);
1042    5    1    14             ELSE PUT EDIT(FILLRATE) (F(5,3));
1043    5    1    14             PUT SKIP EDIT ('III.IMMEDIATE SATISFACTION RATE= ') (COL(44),
                                 A);
1044    5    1    14             IF DEMANDS(I)=0 THEN PUT EDIT ('THERE WERE NO DEMANDS') (A);
1046    5    1    14             ELSE PUT EDIT (DEMSAT(I)/DEMANDS(I)) (F(5,3));
1047    5    1    14             PUT SKIP EDIT('IV.AVERAGE ON HAND STOCK= ',STOCKAREA(I)/STIME)
                                             (COL(45),A,F(5,2));
1048    5    1    14             PUT SKIP EDIT('V.AVERAGE BACKORDERS= ',BOAREA(I)/STIME)
                                             (COL(46),A,F(5,2));
1049    5    1    14          IF BO(I)=0 THEN DUMMY=0;
1051    5    1    14          ELSE DUMMY=BOAREA(I)/BO(I);
1052    5    1    14             PUT SKIP EDIT('VI.AVERAGE BACKORDER TIME= ',DUMMY)
                                             (COL(45),A,F(5,2));
1053    5    1    14             IF AIRCRAFT(I)>0 THEN
1054    5    1    14                DO;
1055    5    2    14                   PUT SKIP EDIT('VII.AVERAGE DOWN ACFT= ',AVGDAC(I))
                                                   (COL(44),A,F(5,2));
1056    5    2    14             IF DAC(I)=0 THEN DUMMY=0;
1058    5    2    14             ELSE
                                    DO;
1059    5    3    14                   DUMMY=0;
1060    5    3    14                   DO J=1 TO TOTSAMPLES;
1061    5    4    14                      DUMMY=DUMMY+DACAREA(I,J);
1062    5    4    14                   END;
1063    5    3    14                   DUMMY=DUMMY/DAC(I);
1064    5    3    14                   END;
1065    5    2    14                   PUT SKIP EDIT('VIII.AVERAGE DOWN ACFT TIME= ',DUMMY)
                                                   (COL(43),A,F(5,2));
1066    5    2    14             ACT_CRIT(I)=AVGDAC(I)/AIRCRAFT(I);
1067    5    2    14          PUT SKIP EDIT ('IX.CRITERION:') (COL(45),A);
1068    5    2    14          PUT SKIP EDIT ('A.MEAN= ',ACT_CRIT(I))(COL(48),A,F(7,5));
1069    5    2    14          PUT SKIP EDIT ('B.STANDARD DEVIATION OF MEAN= ',
                                 SQRT(VARCRITERION(I)/TOTSAMPLES))(COL(48),A,F(7,5));
1070    5    2    14          PUT SKIP EDIT ('C.',(1-ALPHAT)*100,
                                 '% CONFIDENCE INTERVAL ABOUT THE MEAN=') (COL(48),A,F(2),A);
1071    5    2    14          DUMMY=MAX(0,ACT_CRIT(I)-TSTAT*SQRT(VARCRITERION(I)/TOTSAMPLES));
1072    5    2    14          PUT SKIP(2) EDIT(DUMMY,
                                         ' < AVERAGE CRITERION < ',ACT_CRIT(I)+TSTAT*SQRT(
                                 VARCRITERION(I)/TOTSAMPLES)) (COL(49),F(7,5),A,F(7,5));
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1073    5    2    14                    END;
1074    5    1    14                 PUT SKIP(4);
1075    5    1    14                 DO II=1 TO 33;
1076    5    2    14                    PUT EDIT('*') (A);
1077    5    2    14                 END;
1078    5    1    14                 PUT EDIT('N O N - Z E R O   D I S T R I B U T I O N  '||
                                     'V A L U E S') (A);
1079    5    1    14                  DO II=37 TO 120;
1080    5    2    14                    PUT EDIT('*') (A);
1081    5    2    14                 END;
1082    5    1    14           IF SWITCH=0 THEN
1083    5    1    14              DO;
1084    5    2    14                 IF AIRCRAFT(I)=0 THEN
1085    5    2    14                    DO;
1086    5    3    14                       XX=48;
1087    5    3    14                       AA=46;
1088    5    3    14                    END;
1089    5    2    14                 ELSE
                                        DO;
1090    5    3    14                       XX=18;
1091    5    3    14                       AA=16;
1092    5    3    14                    END;
1093    5    2    14                 PUT SKIP(3) EDIT('ON HAND STOCK-BACKORDERS:')(COL(XX),A);
1094    5    2    14                 IF AIRCRAFT(I)>0 THEN PUT EDIT('DOWN ACFT:')(COL(86),A);
1096    5    2    14                 PUT SKIP(2) EDIT('LEVEL       PROPORTION OF TIME')(COL(AA),A);
1097    5    2    14                 IF AIRCRAFT(I)>0 THEN PUT EDIT('|','LEVEL       PROPORTION'||
                                     ' OF TIME') (COL(60),A,COL(76),A);
1099    5    2    14                 BOB=0;
1100    5    2    14                 IF AIRCRAFT(I)=0 THEN DO;
1102    5    3    14                 DUM3=1;
1103    5    3    14                 PP=1;
1104    5    3    14                 END;
1105    5    2    14                 ELSE DO;
1106    5    3    14                 DUM3=0;
1107    5    3    14                 PP=0;
1108    5    3    14                 END;
1109    5    2    14                 DUM1=0;
1110    5    2    14                 DUM2=0;
1111    5    2    14                 NN=MAXSTOCK+1;
1112    5    2    14                 MM=1;
1113    5    2    14                 INDD=0;
1114    5    2    14                 DO WHILE(BOB=0);
1115    5    3    14                 DO WHILE(DUM1=0 & INDD=0);
1116    5    4    14                    IF BODIST(I,NN)>0 THEN DUM1=1;
1118    5    4    14                    ELSE NN=NN-1;
1119    5    4    14                    IF NN=0 | NN=-1 THEN
1120    5    4    14                       DO;
1121    5    5    14                          DUM1=1;
1122    5    5    14                          INDD=1;
1123    5    5    14                          NN=0;
1124    5    5    14                       END;
1125    5    4    14                 END;
1126    5    3    14                 IF INDD=0 THEN IF BODIST(I,NN)>0 THEN
1129    5    3    14                    DO;
1129    5    4    14                       IF NN<=MAXSTOCK THEN PUT SKIP EDIT('-',NN,BODIST(I,NN))
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                      /STIME) (COL(AA+1),A,F(2),X(13),F(5,3));
1131    5    4    14                  ELSE PUT SKIP EDIT('<-',MAXSTOCK,SCDIST(I,NN)/STIME)
                                      (COL(AA),A,F(2),X(13),F(5,3));
1132    5    4    14                  NN=NN-1;
1133    5    4    14                  DUM1=0;
1134    5    4    14                END;
1135    5    3    14            IF INDD=1 & NN=0 THEN
1136    5    3    14                DO:
1137    5    4    14                  IF ZERO>0 THEN
1138    5    4    14                    DO;
1139    5    5    14                  PUT SKIP EDIT('0',ZERO)(COL(AA+3),A,X(13),F(5,3));
1140    5    5    14                  NN=-1;
1141    5    5    14                    END;
1142    5    4    14                  ELSE
                                        DO;
1143    5    5    14                      INDD=2;
1144    5    5    14                      NN=-2;
1145    5    5    14                    END;
1146    5    4    14                END;
1147    5    3    14            DO WHILE(DUM2=0 & INDD=2);
1148    5    4    14                IF STOCKDIST(I,MM)>0 THEN DUM2=1;
1150    5    4    14                ELSE MM=MM+1;
1151    5    4    14                IF MM>MAXSTOCK+1 THEN
1152    5    4    14                  DO;
1153    5    5    14                    INDD=3;
1154    5    5    14                    DUM2=1;
1155    5    5    14                  END;
1156    5    4    14            END;
1157    5    3    14            IF INDD=2 THEN
1158    5    3    14                DO;
1159    5    4    14                    IF MM<=MAXSTOCK THEN PUT EDIT('+',MM,STOCKDIST(I,MM)/
                                        STIME) (COL(AA+1),A,F(2),X(13),F(5,3));
1161    5    4    14                    ELSE PUT EDIT('>+',MAXSTOCK,STOCKDIST(I,MM)/STIME)
                                        (COL(AA),A,F(2),X(13),F(5,3));
1162    5    4    14                  IF MM<=MAXSTOCK THEN DO;
1164    5    5    14                    MM=MM+1;
1165    5    5    14                    DUM2=0;
1166    5    5    14                END;
1167    5    4    14                  END;
1168    5    3    14            DO WHILE(DUM3=0);
1169    5    4    14                IF PP<=AIRCRAFT(I) THEN
1170    5    4    14                  DO;
1171    5    5    14                    IF DACDIST(I,PP)>0 THEN DUM3=1;
1173    5    5    14                    ELSE PP=PP+1;
1174    5    5    14                  END;
1175    5    4    14                ELSE DUM3=1;
1176    5    4    14            END;
1177    5    3    14            IF PP<=AIRCRAFT(I) THEN IF DACDIST(I,PP)>0 THEN
1179    5    3    14                DO;
1180    5    4    14                  IF DUM1=1 & DUM2=1 THEN PUT SKIP;
1182    5    4    14                  PUT EDIT(PP,DACDIST(I,PP)/STIME)(COL(77),F(2),
                                      X(14),F(5,3));
1183    5    4    14                  PP=PP+1;
1184    5    4    14                  DUM3=0;
1185    5    4    14                END;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1186    5    3    14                IF NN=-1 THEN
1187    5    3    14                   DO;
1188    5    4    14                      INDD=2;
1189    5    4    14                      NN=-2;
1190    5    4    14                   END;
1191    5    3    14                IF DUM1=1 & DUM2=1 & DUM3=1 THEN BBB=1;
1193    5    3    14                ELSE
                                       DO;
1194    5    4    14                      IF AIRCRAFT(I)>0 THEN PUT SKIP(0) EDIT('|') (COL(60),A);
1196    5    4    14                   END;
1197    5    3    14                END;
1198    5    2    14             END;
1199    5    1    14          ELSE
                                 DO;
1200    5    2    14                IF AIRCRAFT(I)=0 THEN
1201    5    2    14                   DO;
1202    5    3    14                      XX=24;
1203    5    3    14                      YY=85;
1204    5    3    14                      AA=16;
1205    5    3    14                      WW=1;
1206    5    3    14                      BB=60;
1207    5    3    14                      CC=15;
1208    5    3    14                   END;
1209    5    2    14                ELSE
                                       DO;
1210    5    3    14                      XX=14;
1211    5    3    14                      YY=55;
1212    5    3    14                      AA=6;
1213    5    3    14                      BB=40;
1214    5    3    14                      CC=6;
1215    5    3    14                      WW=2;
1216    5    3    14                   END;
1217    5    2    14                PUT SKIP(3) EDIT('ON HAND STOCK:') (COL(XX),A);
1218    5    2    14                PUT EDIT('BACKORDERS:') (COL(YY),A);
1219    5    2    14                IF AIRCRAFT(I)>0 THEN PUT EDIT('DOWN ACFT:')(COL(96),A);
1221    5    2    14                PUT SKIP(2) EDIT('LEVEL      PROPORTION OF TIME')(COL(AA),A);
1222    5    2    14                BBB=0;
1223    5    2    14                   DO II=1 TO WW;
1224    5    3    14                      PUT EDIT('|','LEVEL      PROPORTION OF TIME')
                                          (COL(BB+BBB),A,COL(BB+BBB+CC),A);
1225    5    3    14                      BBB=40;
1226    5    3    14                   END;
1227    5    2    14                BOB=0;
1228    5    2    14                IF AIRCRAFT(I)=0 THEN DO;
1230    5    3    14                DUM3=1;
1231    5    3    14                PP=1;
1232    5    3    14                END;
1233    5    2    14                ELSE DO;
1234    5    3    14                DUM3=0;
1235    5    3    14                PP=0;
1236    5    3    14                END;
1237    5    2    14                DUM1=0;
1238    5    2    14                DUM2=0;
1239    5    2    14                NN=0;
1240    5    2    14                MM=0;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1241    5    2    14          DO WHILE(BOB=0);
1242    5    3    14          DO WHILE(DUM1=0);
1243    5    4    14             IF NN<=MAXSTOCK+1 THEN
1244    5    4    14                DO;
1245    5    5    14                   IF STOCKDIST(I,NN)>0 THEN DUM1=1;
1247    5    5    14                   ELSE NN=NN+1;
1248    5    5    14                END;
1249    5    4    14             ELSE DUM1=1;
1250    5    4    14          END;
1251    5    3    14          IF NN<=MAXSTOCK+1 THEN IF STOCKDIST(I,NN)>0 THEN
1253    5    3    14             DO;
1254    5    4    14                IF NN<=MAXSTOCK THEN
1255    5    4    14                PUT SKIP EDIT(NN,STOCKDIST(I,NN)/STIME)(COL(CC+1),
                                   F(2),X(14),F(5,3));
1256    5    4    14                ELSE PUT SKIP EDIT('>',MAXSTOCK,STOCKDIST(I,NN)/STIME)
                                   (COL(CC),A,F(2),X(14),F(5,3));
1257    5    4    14                NN=NN+1;
1258    5    4    14                DUM1=0;
1259    5    4    14             END;
1260    5    3    14          DO WHILE(DUM2=0);
1261    5    4    14             IF MM<=MAXSTOCK+1 THEN
1262    5    4    14                DO;
1263    5    5    14                   IF BODIST(I,MM)>0 THEN DUM2=1;
1265    5    5    14                   ELSE MM=MM+1;
1266    5    5    14                END;
1267    5    4    14             ELSE DUM2=1;
1268    5    4    14          END;
1269    5    3    14          IF MM<=MAXSTOCK+1 THEN IF BODIST(I,MM)>0 THEN
1271    5    3    14             DO;
1272    5    4    14                IF DUM1=1 THEN PUT SKIP;
1274    5    4    14                IF MM<=MAXSTOCK THEN
1275    5    4    14                PUT EDIT(MM,BODIST(I,MM)/STIME)(COL(BB+CC+1),F(2),X(14),
                                   F(5,3));
1276    5    4    14                ELSE PUT EDIT('>',MAXSTOCK,BODIST(I,MM)/STIME)
                                   (COL(BB+CC),A,F(2),X(14),F(5,3));
1277    5    4    14                MM=MM+1;
1278    5    4    14                DUM2=0;
1279    5    4    14             END;
1280    5    3    14          DO WHILE(DUM3=0);
1281    5    4    14             IF PP<=AIRCRAFT(I) THEN
1282    5    4    14                DO;
1283    5    5    14                   IF DACDIST(I,PP)>0 THEN DUM3=1;
1285    5    5    14                   ELSE PP=PP+1;
1286    5    5    14                END;
1287    5    4    14             ELSE DUM3=1;
1288    5    4    14          END;
1289    5    3    14          IF PP<=AIRCRAFT(I) THEN IF DACDIST(I,PP)>0 THEN
1291    5    3    14             DO;
1292    5    4    14                IF DUM1=1 & DUM2=1 THEN PUT SKIP;
1294    5    4    14                PUT EDIT(PP,DACDIST(I,PP)/STIME)(COL(87),F(2),
                                   X(14),F(5,3));
1295    5    4    14                PP=PP+1;
1296    5    4    14                DUM3=0;
1297    5    4    14             END;
1298    5    3    14          IF DUM1=1 & DUM2=1 & DUM3=1 THEN BOB=1;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1300    5    3    14                   ELSE
                                          DO:
1301    5    4    14                         IF AIRCRAFT(I)>0 THEN CHR='|';
1303    5    4    14                         ELSE CHR='';
1304    5    4    14                         PUT SKIP(0) EDIT('|',CHR)(COL(38),A,X(39),A);
1305    5    4    14                      END;
1306    5    3    14                   END;
1307    5    2    14                      PUT SKIP(5) EDIT('*** PORT''S CARRIERS ***')(COL(50),A);
1308    5    2    14                      PUT SKIP(1);
1309    5    2    14                         DO J=1 TO TOTNUMBASES:
1310    5    3    14                            IF PORT(J)=I THEN PUT SKIP EDIT(BASENAME(J))
                                                (COL(54),A);
1312    5    3    14                         END;
1313    5    2    14                END;
1314    5    1    14                   PUT SKIP(2);
1315    5    1    14                   DO II=1 TO 120;
1316    5    2    14                      PUT EDIT('-') (A);
1317    5    2    14                   END;
1318    5    1    14                   IF I=TOTNUMBASES THEN
1319    5    1    14                   DO;
1320    5    2    14                   PUT PAGE;
1321    5    2    14                   PUT SKIP(2) EDIT('SYSTEM PERFORMANCE') (COL(49),A);
1322    5    2    14                   PUT SKIP(3) EDIT('BASE','|','WEIGHT','|','ACTUAL CRITERION',
                                       '|','DESIRED CRITERION','|','DEVIATION') (COL(5),A,COL(25),A,
                                       COL(34),A,COL(49),A,COL(54),A,COL(73),A,COL(77),A,COL(97),A,
                                       COL(103),A);
1323    5    2    14                   SYSPERF=0; SLACKPERF=0;
1325    5    2    14                   SUMDAC=0; TOTAC=0;
1327    5    2    14                   DO II=1 TO TOTNUMBASES;
1328    5    3    14                      IF AIRCRAFT (II)>0 THEN DO:
1330    5    4    14                      SUMDAC=SUMDAC+AVGDAC(II);
1331    5    4    14                      TOTAC=TOTAC+AIRCRAFT(II);
1332    5    4    14                      IF ACT_CRIT(II)>DES_CRIT(II) THEN CHR='+';
1334    5    4    14                      ELSE IF ACT_CRIT(II)<DES_CRIT(II) THEN CHR='-';
1336    5    4    14                      ELSE CHR=' ';
1337    5    4    14                      PUT SKIP EDIT(II,')',BASENAME(II),'|',WEIGHT(II),'|',
                                          ACT_CRIT(II),'|',DES_CRIT(II),'|',CHR,ABS(ACT_CRIT(II)-
                                             DES_CRIT(II))) (F(3),A,A,COL(25),A,COL(35),F(4,1),
                                          COL(49),A,COL(59),F(5,3),COL(73),A,COL(83),F(5,3),COL(97),
                                          A,COL(105),A,F(5,3));
1338    5    4    14                      SYSPERF=SYSPERF+MAX(0,ACT_CRIT(II)-DES_CRIT(II))*
                                          WEIGHT(II)/SUMWEIGHT;
1339    5    4    14                      END;
1340    5    3    14                   END;
1341    5    2    14                   PUT SKIP(2) EDIT('SYSTEM PERFORMANCE= ',SUMDAC/TOTAC)
                                       (COL(42),A,F(5,3));
1342    5    2    14                   PUT SKIP EDIT('WEIGHTED AVERAGE PERFORMANCE= ',SYSPERF)
                                       (COL(42),A,F(5,3));
1343    5    2    14                   PUT SKIP(5);
1344    5    2    14                      DO II=1 TO 120;
1345    5    3    14                         PUT EDIT('*') (A);
1346    5    3    14                      END;
1347    5    2    14                   END;
[illegible] 5    1    14                   ELSE PUT PAGE;
[illegible] 5    1    14                   END;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1350    5          14              GO TO FINISH;
1351    5          14          END REPORT;


                                     /*****************************************************************/
                                     /*DISTRIBUTIONS:                                                 */
                                     /*DISTRIBUTIONS UPDATES TIME WEIGHTED DISTRIBUTIONS FOR STOCK    */
                                     /*LEVELS, BACKORDER LEVELS AND DOWNED AIRCRAFT LEVELS.  IT ALSO  */
                                     /*MONITORS THE TIME DURING WHICH A PORT IS IN BOTH THE POSITIVE  */
                                     /*BACKORDER AND POSITIVE STOCK STATE SIMULTANEOUSLY.  DISTRIBU-  */
                                     /*TIONS IS CALLED WHENEVER THE APPROPRIATE LEVELS CHANGE AS A    */
                                     /*RESULT OF SIMULATION ACTIVITY.                                 */
                                     /*****************************************************************/

1352    4           4            DISTRIBUTIONS:PROC(BASE,DIST);
1353    5          15              DCL BASE BIN FLOAT;
1354    5          15              DCL DIST CHAR(*);
1355    5          15              IF CLOCK<= RUNIN THEN RETURN;
1357    5          15              IF DIST='STOCK' THEN
1358    5          15                 DO;
1359    5     1    15                    IF STOCKLEVEL(BASE)<=MAXSTOCK THEN STOCKDIST(BASE,STOCKLEVEL(
                                         BASE))=STOCKDIST(BASE,STOCKLEVEL(BASE))+CLOCK-LSTIME(BASE);
1361    5     1    15                    ELSE STOCKDIST(BASE,MAXSTOCK+1)=STOCKDIST(BASE,MAXSTOCK+1)+
                                         CLOCK-LSTIME(BASE);
1362    5     1    15                    STOCKAREA(BASE)=STOCKAREA(BASE)+STOCKLEVEL(BASE)*(CLOCK-
                                         LSTIME(BASE));
1363    5     1    15                    LSTIME(BASE)=CLOCK;
1364    5     1    15                 END;
1365    5          15              ELSE IF DIST='BO' THEN
1366    5          15                 DO;
1367    5     1    15                    IF BACKORDERS(BASE)<=MAXSTOCK THEN
1368    5     1    15                    BODIST(BASE,BACKORDERS(BASE))=BODIST(BASE,BACKORDERS(BASE))+
                                         CLOCK-LBOTIME(BASE);
1369    5     1    15                    ELSE BODIST(BASE,MAXSTOCK+1)=BODIST(BASE,MAXSTOCK+1)+CLOCK
                                         -LBOTIME(BASE);
1370    5     1    15                    BOAREA(BASE)=BOAREA(BASE)+BACKORDERS(BASE)*(CLOCK
                                         -LBOTIME(BASE));
1371    5     1    15                    LBOTIME(BASE)=CLOCK;
1372    5     1    15                 END;
1373    5          15              ELSE IF DIST='DAC' THEN
1374    5          15                 DO;
1375    5     1    15                    DACDIST(BASE,DACNUM(BASE))=DACDIST(BASE,DACNUM(BASE))+
                                         CLOCK-LDACTIME(BASE);
1376    5     1    15                    DACAREA(BASE,NUMSAMPS)=DACAREA(BASE,NUMSAMPS)+DACNUM(BASE)*
                                         (CLOCK-LDACTIME(BASE));
1377    5     1    15                    LDACTIME(BASE)=CLOCK;
1378    5     1    15                 END;
1379    5          15            END DISTRIBUTIONS;

1380    4           4              END B3;


                                     /*****************************************************************/
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

                                     /*AVAILQ:                                                        */
                                     /*AVAILQ, WITH PARAMETERS (W,X,Y,Z), MAKES AN ENGINE AVAILABLE   */
                                     /*AT BASE W FROM BASE X IN TIME Z FROM THE PRESENT.  IF PARA-    */
                                     /*METER Y IS POSITIVE, THIS INDICATES ENGINE TRANSPORT FROM A    */
                                     /*CARRIER AND ACCOUNTS FOR THE TIME BEFORE THE CARRIER DOCKS AT  */
                                     /*PORT.  ONCE THE TIME OF AVAILABILITY IS DETERMINED, THE        */
                                     /*AVAILABILITY EVENT IS QUEUED (IN INCREASING ORDER OF TIME)     */
                                     /*AS A FUTURE EVENT.                                             */
                                     /****************************************************************/

1381   3           3               AVAILQ:PROCEDURE(BASETO,BASEFROM,SHIPINDEX,REPTIME);
1382   4          16                 DCL(BASETO,BASEFROM,SHIPINDEX,REPTIME) BIN FLOAT;
1383   4          16                 IF SHIPINDEX¬=0
1384   4          16                     THEN ENGAVAILTIME(ENGAVAILQMT)=TRAVTIME(BASEFROM,BASETO)+
                                              REPTIME+SHIPTIME(SHIPINDEX);
1385   4          16                     ELSE IF REPTIME=0
1386   4          16                         THEN ENGAVAILTIME(ENGAVAILQMT)=TRAVTIME(BASEFROM,
                                                  BASETO)+CLOCK;
1387   4          16                         ELSE ENGAVAILTIME(ENGAVAILQMT)=TRAVTIME(BASEFROM,BASETO)
                                              +REPTIME+CLOCK;
1388   4          16                 BASENGAVAIL(ENGAVAILQMT)=BASETO;
1389   4          16                 ENGAVAILQDUM=ENGAVAILQBOT;
1390   4          16                 DO WHILE (ENGAVAILTIME(ENGAVAILQMT)<ENGAVAILTIME(ENGAVAILQDUM));
1391   4     1    16                 ENGAVAILQDUM=ENGAVAILQBPT(ENGAVAILQDUM);
1392   4     1    16                 END;
1393   4          16                 ENGAVAILQHOLD=ENGAVAILQFPT(ENGAVAILQMT);
1394   4          16                 IF ENGAVAILQDUM=ENGAVAILQBOT
1395   4          16                     THEN DO;
1396   4     1    16                         ENGAVAILQFPT(ENGAVAILQBOT)=ENGAVAILQMT;
1397   4     1    16                         ENGAVAILQFPT(ENGAVAILQMT)=0;
1398   4     1    16                 ENGAVAILQBPT(ENGAVAILQMT)=ENGAVAILQBOT;
1399   4     1    16                         ENGAVAILQBOT=ENGAVAILQMT;
1400   4     1    16                     END;
1401   4          16                     ELSE DO;
1402   4     1    16                         ENGAVAILQFPT(ENGAVAILQMT)=ENGAVAILQFPT(ENGAVAILQDUM);
1403   4     1    16                         ENGAVAILQFPT(ENGAVAILQDUM)=ENGAVAILQMT;
1404   4     1    16                         ENGAVAILQBPT(ENGAVAILQMT)=ENGAVAILQDUM;
1405   4     1    16                         ENGAVAILQBPT(ENGAVAILQFPT(ENGAVAILQMT))=ENGAVAILQMT;
1406   4     1    16                     END;
1407   4          16                 ENGAVAILQMT=ENGAVAILQHOLD;
1408   4          16                 NJUNAVAILTIME=ENGAVAILTIME(ENGAVAILQFPT(0));
1409   4          16                 END AVAILQ;
1410   3           3                 END B2;
1411   2           2                 END B1;


                                     /****************************************************************/
                                     /*RNG:                                                           */
                                     /*RNG RETURNS AN EXPONENTIALLY DISTRIBUTED RANDOM VARIATE        */
                                     /*GIVEN A MEAN, AND A UNIFORM 0-1 RANDOM SEED.                   */
                                     /****************************************************************/

1412   1           1               RNG:PROC(MEAN,U) RETURNS(BIN FLOAT);
1413   2          17                 DCL(MEAN,U) BIN FLOAT;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1414    2          17           U=RAND(U);
1415    2          17           RETURN(-MEAN*LOG(U));
1416    2          17         END RNG;


                                /*****************************************************************/
                                /*SHIPTIME_RNG:                                                  */
                                /*SHIPTIME_RNG RETURNS A NORMALLY DISTRIBUTED RANDOM VARIATE     */
                                /*GIVEN A MEAN, STANDARD DEVIATION AND A RANDOM NUMBER SEED.     */
                                /*****************************************************************/

1417    1           1         SHIPTIME_RNG:PROC(MU,SIGMA,U) RETURNS(BIN FLOAT);
1418    2          18           DCL(MU,SIGMA,U,I,L,Z,LASTU,X,Y,N) BIN FLOAT;
1419    2          18           I=1; L=2;
1421    2          18           U=RAND(U);
1422    2          18           Z=2*U;
1423    2          18              DO WHILE(Z<1);
1424    2    1     18                 I=I+1;
1425    2    1     18                 Z=Z*2;
1426    2    1     18              END;
1427    2          18           LASTU=0;
1428    2          18           U=RAND(U);
1429    2          18              DO WHILE(MOD(L,2)=0);
1430    2    1     18                 U=(U-LASTU)/(1-LASTU);
1431    2    1     18                 X=.5**I+(.5**(I+1)-.5**I)*U;
1432    2    1     18                 Y=.5*(X**2-.5**(2*I));
1433    2    1     18                 U=RAND(U);
1434    2    1     18                 L=1;
1435    2    1     18                    DO WHILE(Y>U);
1436    2    2     18                       L=L+1;
1437    2    2     18                       LASTU=U;
1438    2    2     18                       U=RAND(U);
1439    2    2     18                    END;
1440    2    1     18              END;
1441    2          18           U=RAND(U);
1442    2          18           IF U<=.5 THEN N=X*SIGMA+MU;
1444    2          18           ELSE N=-X*SIGMA+MU;
1445    2          18           IF N>0 THEN RETURN(N);
1447    2          18           ELSE RETURN(MU);
1448    2          18         END SHIPTIME_RNG;


                                /*****************************************************************/
                                /*SAFETY_STOCK:                                                  */
                                /*SAFETY_STOCK GENERATES A SAFETY STOCKLEVEL, GIVEN A CRITICAL   */
                                /*OUT OF STOCK PROBABILITY.                                      */
                                /*****************************************************************/

1449    1           1         SAFETY_STOCK:PROC(LAMDA,C) RETURNS(BIN FLOAT);
1450    2          19           DCL(LAMDA,C,P,CUMP,S) BIN FLOAT;
1451    2          19           P=EXP(-LAMDA);
1452    2          19           CUMP=P;
1453    2          19           S=0;
```

```
STMT LEVEL NEST BLOCK MLVL  SOURCE TEXT

1454    2           19               DO WHILE(CUMP<=C);
1455    2     1     19                  S=S+1;
1456    2     1     19                  P=P*LAMDA/S;
1457    2     1     19                  CUMP=CUMP+P;
1458    2     1     19               END;
1459    2           19            RETURN(S);
1460    2           19           END SAFETY_STOCK;


1461    1            1            FINISH:
                                 END NAVY;
```

ERRORS/WARNINGS DETECTED DURING CODE GENERATION:

WARNING: NO FILE SPECIFIED. SYSIN/SYSPRINT ASSUMED. (CG0C)
WARNING: PL/C BUILT-IN FUNCTION USED. (CG13)

APPENDIX B: User Instructions for Data Input

Through the specification of the input values, many options in the system structure, performance and control are available.

Input information is in the form of a stream of data cards. Data can appear in any column but a blank must follow each piece of data. Quotation marks are required for some data inputs (shown on example card). The data cards must be in the order given below.

Instructions for coding along with an example 'coded' card are provided below:

Data Set #1: System Attributes

(one card)

'J52P'	'NARF JAX'	16	3	296	6	2	2	30	30
(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)

(a) Engine model type for which system is being simulated (no more than 20 characters long).

(b) Depot name (no more than 20 characters long).

(c) Total number of bases.

(d) Total number of carriers, if Twisted Pipeline Configuration is desired:

"0" if Continuous Pipeline Configuration is desired.

(e) Total number of aircraft (number of aircraft at some bases may not be an integer and should be rounded up when figuring total number of aircraft in system).

(f) Total number of aircraft type.

(g) Total number of NAVY combination codes associated with system.

(h) The maximum engine capacity of all aircraft in system.

(i) The maximum schedule length of all carriers (carrier schedules are discussed in the explanation of Data Set #6), in Twisted Pipeline Configuration.

"0" in Continuous Pipeline Configuration

(j) Total number of samples. This quantity is used for statistical analysis purposes and must be a number divisible by 2. The larger the value entered, the higher the confidence in the simulation output data along with a higher cost associated with a longer computer run. This quantity should be at least 20 and a value of 30 is usually adequate.

Data Set #2: Base Random Number Seeds

(As many cards as needed)

.09225305	.165307001	• • • • • • •	.059315127	• • •
(a)	(b)	• • • • • • •	(k)	• • •

(a) Random number seed for base #1.

(b) Random number seed for base #2.

•
•
•

(k) Random number seed for base #k.

•
•
•

k = 1, 2, 3, • • • , total number of bases

Each value entered must be a 9-digit number between "0" and "1" with an odd last digit.

Data Set #3: Analysis Parameters
(one card)

90	2	0	1.65	.05	2.045	.05
(a)	(b)	(c)	(d)	(e)	(f)	(g)

Before any statistical analysis can be performed on the observed values for each base's 'criterion' (= average down aircraft/total number of aircraft), the variances of these observed quantities must be estimated. The problem arises from the fact that the data are serially correlated (that is, time dependent). In other words, if we let $x_i(t)$ = number of down aircraft at base i at time t, then $x_i(t + s)$ could be dependent on $x_i(t)$ and $x_i(t)$ could be dependent on $x_i(t - s)$ and so on. The simple formulas for computing sample variances are based on independent observations. It would seem reasonable to say that there may exist a value 's' (sometimes very large) such that $x_i(t + s)$ is independent of $x_i(t)$. If we could find such a value 's' and then took observations 's' time units apart, the values obtained would be independent and the simple

variance formulas would be applicable.

For a given value of 's,' SPAERS tests the null hypothesis for each base $\underline{i}$, that $x_i(t_1)$, $x_i(t_2)$, $x_i(t_3)$, $\cdots$, $x_i(t_n)$ is a random stream (i.e., $x_i(t_j)$'s are independent) where $t_{j+1} - t_j = s$ for $j = 1, 2, \cdots, N-1$ and N = total number of samples specified on Data Set #1. The test of independence is said to have 'failed' if the null hypothesis is rejected.

The technique employed in testing the null hypothesis and choosing a new 's' if the test fails is outlined in Appendix D.

(a) Initial length of sample interval. This quantity represents the user's guess to the value of 's.' If the test of independence using this value of 's' fails, then 's' is increased automatically in the program, more data are collected, and the new 's' is tested for independence. This procedure is repeated until the test of independence passes or the limit of tests performed is reached (item (b) on this card). An initial guess would be to set 's' equal to twice the maximum depot cycle time of all bases. Since depot cycle time is typically the longest, the system performance is probably most affected by depot activity. Thus, by letting s = 2 • maximum depot cycle time, we are allowing for the system to 'flush' itself out.

(b) The maximum number of independence tests to be performed. SPAERS is constructed so that the total number of samples specified on Data Set #1 is constant no matter how many tests of independence are performed. For each successive test performed, sample interval is actually doubled (see Appendix D). Thus, actual simulation time ≈ Run-in period + total number of samples x initial sample interval (item (a) above) x 2^{k-1} days, where $\underline{k}$ is the number of tests performed. The purpose of including this quantity is to provide an upper bound and hence a stopping criterion (in the interest of computer time) if independence may not be found within a reasonable number of times.

If twice the depot cycle time is used for the

initial sample interval, then more than 3 independence tests should not be executed without further investigation into the reasons for 'high dependence.'

In general, if a small initial sample interval is chosen (< 2 x depot cycle time), there is a higher probability of rejecting the null hypothesis initially, so more independence tests should be allowed.

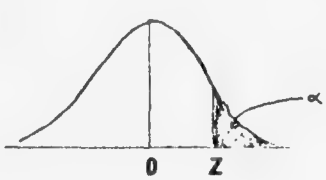
(c) 0, if all bases are required to pass independence test.

Base number of a 'critical' base which is believed to be the highest candidate for failure of independence. In choosing a 'critical' base it is the belief that if this base passes the test for independence, then it is reasonable that the remaining bases would also pass the test. Thus, it is only necessary to require that the independence test pass at the 'critical' base.

(d) and (e) Parameters for the test of independence.

Let $\underline{z}$ denote the value entered in (d) and α the value entered in (e).

Then, $\underline{z}$ is the $1 - \alpha$ significance point of the standard normal distribution.
Graphically:



$\underline{Z}$ is the test statistic and α is the significance level for testing the null hypothesis, H_o, that $x(t_1)$, $x(t_2)$, • • • , $x(t_n)$ are independent.

First, the user must choose a significance level α. We note that:

α = Probability (rejecting H_o when H_o is true). A small α level means that there is a higher probability of accepting H_o when it is false and would lead to an underestimation of the variance of $\bar{x}_N$. On the other hand, a large α level means that there is a higher probability of rejecting H_o when it is true and would lead to a less efficient estimate of the confidence

interval of $\overline{x}_N$.* Significance levels of 0.05 and 0.10 are typically used. Levels smaller than 0.025 and larger than 0.25 should be avoided.

$\underline{Z}$ is obtained from standard normal tables such that: Probability (standard normal $\leq$ Z) = 1 - α.

It would seem reasonable to accept the hypothesis of independence for all bases if: the number of bases that failed the test of independence divided by the total number of bases tested is less than α, since there is an error 'α' of rejecting H_0. This is in fact the passing criterion for system independence used by SPAERS.

(f) and (g) Confidence interval parameters.

Let $\underline{t}$ denote the value entered in (f) and γ the value entered in (g).

$\underline{t}$ and γ are used to compute confidence intervals on the mean criterion at each base.

Interpretation of $\underline{t}$ and γ is as follows:

$$(1)\quad \Pr\{L_1 \leq \mu \leq L_2\} = 1 - \gamma,$$

where: $L_1 = \overline{x} - \dfrac{ts}{\sqrt{n}}$

$L_2 = \overline{x} + \dfrac{ts}{\sqrt{n}}$

μ = true mean

and, $\overline{x}$ = sample (observed mean)

s = sample standard deviation

n = number of samples observed

$t = 100(1 - \frac{\gamma}{2})$ percentile of the student's t-distribution with $n - 1$ degrees of freedom.**

In short, equation (1) defines limits L_1 and L_2 such that there is 100 $(1 - \gamma)$% confidence that the true mean lies between these bounds.

Commonly used values for γ are 0.05 and 0.01 to give

---

* Reference [3], page 7.

** Reference [4].

a "confidence interval" of 95% and 99%, respectively.

Once a γ level has been chosen, a t value may be obtained from t-distribution tables. t is chosen such that:

$$Pr\{x \le t\} = 1 - \frac{\gamma}{2}$$

where x is a student's t-distributed random variable with n - 1 degrees of freedom. (n = total number samples; entered on Data Set #1.)

Data Set #4: System Combination Codes

(As many cards as needed)

102	104	• • •	108	• • •
(a)	(b)	• • •	(k)	• • •

(a) Navy combination code. Throughout the remainder of this section and in the simulation, CC102 will be indexed by a "1."

(b) Navy combination code. Throughout the remainder of this section and in the simulation, CC104 will be indexed by a "2."

•
•
•

(k) Navy combination code. Throughout the remainder of this section and in the simulation, CC108 will be indexed by a "k."

•
•
•

k = 1, 2, 3, • • • , total number of combination codes in system

Data Set #5: Aircraft Type Attributes

(one card for each aircraft type)

'AGE'	2	2
(a)	(b)	(c)

(a) Navy aircraft type name (no more than 7 characters long).

(b) Engine capacity.

(c) Combination Code index (see Data Set #4). In this case the combination code of AGE is 104.

For the remainder of this section and in the simulation, an aircraft type will be indexed by the order number in which it was 'inputed.'

Data Set #6: Carrier Attributes

(one card(s) for each carrier)

0	0	0	1	4	5	15	20	60	10	30	10	15
(a)	(b)	(c)	(d)	(e)	(f)							

(a) Mean 'docking' time (time spent in port) in days, if probabilistic times are desired;

0, if deterministic 'docking' times are desired.

(b) Mean 'deployment' time (time spent at sea) in days, if probabilistic time are desired;

0, if deterministic 'deployment' times are desired.

(c) Standard deviation (in days) of docking and deployment time is probabilistic times are desired;

0, if deterministic docking and deployment times are desired.

(d) 0, if probabilistic carrier times are desired. Docking and deployment times will thus be random variables, normally distributed with mean (a) and (b) and standard deviation (c).

1, if deterministic schedules are desired.

(e) 30, if (d) is equal to 0;

The desired number of times before deterministic schedule is repeated, if (d) is equal to 1.

The two numbers in (e) have different meaning and should not be confused. In the probabilistic case, we need to know the future docking and deployment times when determining a failure time for an aircraft since a failure during the 'docking' state is not permitted (aircraft does not fly). Thus, the number entered in this case indicates that we may be safe in not scheduling a failure during the 'docking' state if this number of future docking and deployment times are known. The number 30 is sufficient for MTBR (mean time between removal) = 226 but should be increased if a MTBR much larger than 226 is used.

On the other hand, the deterministic schedule repeats in cycles and thus knowledge of future carrier times is present at all times. The number entered here thus, designates the schedule cycle length or in other words, the number of schedules in the deployment (docking) cycle.

A probabilistic schedule may be desired when a particular carrier has unusual but consistent and/or required deployment and docking times, as may be the case during wartime.

(f) a blank, i.e., no numbers entered, if (d) equals 0;

Docking time of first schedule,
Deployment time of first schedule,
.
.
.
Docking time of ith schedule,
Deployment time if ith schedule, where i = 1, 2, . . . , schedule length (value entered in (e)), if (d) equals 1.
(Note: docking time = time spent in port; deployment time = time spent at sea.)

The order of the cards of Data Set #6 should be the same as the ship numbers of the carriers (see Data Set #10 for ship numbers).

(Important: Data Set #6 is omitted if item (d) on Data Set #1 equals 0, i.e., Continuous Pipeline Configuration.)

Data Set #7: Mean Time Between Removal (MTBR's)

(As many cards as needed)

187	226	• • •	342	• • •
(a)	(b)	• • •	(k)	• • •

(a) MTBR associated with the combination code indexed by 1 (see Card Set #4).

(b) MTBR associated with the combination code indexed by 2.

•
•
•

(k) MTBR associated with the combination code indexed by k.

•
•
•

k = 1, 2, • • • , total number of combination codes

Data Set #8: Combination Code Maintenance Probabilities

(one card for each combination code)

.116	.362	.230	.115	.117
(a)	(b)	(c)	(d)	(e)

Probability that, given an engine failure, the degree of maintenance is:

(a) 3rd degree

(b) 2nd degree

(c) 1st degree

(d) (-1)st degree (maintenance performed at depot on a 1st degree failed engine)

(e) overhaul or 0^{th} degree.

The order of the cards of Data Set #8 should correspond to the index of the respective combination code(s).

Data Set #9: Engine Repair Times

(one card)

19	19	24	23	30
(a)	(b)	(c)	(d)	(e)

The repair time (in days) required for the following

degree maintenance:

(a) 3rd degree

(b) 2nd degree

(c) 1st degree

(d) (-1)st degree

(e) overhaul or 0^{th} degree.

Data Set #10: Base Synopsis

(one data set for each base)

Data Set #10A: Base Attributes

(one card)

'CHERRY POINT'	4	0	0	2	0	1	3	1	.01	2
(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)

(a) Base name (cannot exceed 20 characters in length).

(b) Base number or the order the base's data set. In this example, CHERRY POINT is the 4th base to be read.

(c) Carrier number or the order in which it is read in comparison to other carriers, if the base is a carrier and Twisted Pipeline Configuration is desired;

0, if the base is not a carrier or if the base is a carrier but a Continuous Pipeline Configuration is desired.

(d) 1, if base is an 'acting' port;

0, otherwise.

'Acting' port is defined as a port which serves as an intermediary point for all engine transactions between its carriers and the appropriate higher echelon maintenance bases.

This concept was introduced to allow for the inclusion of a base which neither supports flying activity nor maintenance activity but serves as a port to one or several carriers. This type of base, in many cases, is currently ignored in the analytical models. The reason for this is that the pipeline length from a carrier to base *i*, say, is equal to the sum of the pipeline length from a carrier to its port and the pipeline length from that port to base *i*. The pipeline lengths in a continuous pipeline configuration remain constant and since this port supports neither flying nor maintenance activity, it serves no computational purpose. On the other hand, in the real world system this type of port does serve as an intermediary point and although its exclusion may have no apparent effect in the analytical model, the potential hazards in a twisted pipeline configuration are worth noting.

Excluding this port in a twisted pipeline configuration would mean to consider the carrier's next echelon maintenance base as its port. This would mean that this 'artificial' base has accessibility to the carrier engines that were in repair at the time of a carrier departure from port but have become available as serviceable on-hand stock

since. This accessibility would last until the time of the carrier's next deployment (i.e., the time remaining in present deployment plus the time spent in port). Furthermore, the fact that this base employs the carrier engines increases the probability that the carrier will depart with backorders. These factors will, of course, affect the performance of the system, especially if this base serves as the maintenance activity for many other bases.

Even in the continuous pipeline configuration, the artificial port should be avoided, since increasing the spare stock level at this port to compensate for poor carrier performance would have similar effects to that of the twisted pipeline configuration: this port would have accessibility to these engines when not in demand from the carrier.

It is the belief of the authors that all "real" ports should be included whether continuous or twisted pipeline configurations are desired since their inclusion is more representative of the real-word system. These "real" ports would, as noted, have a "1" under item (d) (a "0" would be entered for artificial ports).

(e) number of aircraft type.

(f) base number of the base which serves as a port for this base, if this base is a carrier <u>and</u> Twisted Pipeline Configuration is being employed;

0, if base is not a carrier <u>or</u> Continuous Pipeline Configuration is being employed.

(g) 0, if it is desired to have the program compute spare

stock level for this base, using the DODI 4230.4 calculation;

1, if it is desired to 'input' the spare stock level.

(h) 0, if (g), above, is at level 0.

Desired spare stock level, if (g), above, is at level 1.

(i) A desired probability of having no outstanding backorders at an arbitrary point in time, if (g), above, is at level 0.

0, if (g), above, is at level 1.

(j) Desired criterion.

(k) Weight of importance relative to other bases. (A desired criterion of 0.01 at base *i* and 0.02 at base *j* would mean that base *i* is seen as being twice as important as base *j* and consequently the weight of base *i* would be twice that of base *j*.)

Data Set #10B: Base Maintenance Activity

(one card)

7	7	6	8	8
(a)	(b)	(c)	(d)	(e)

Maintenance activity (base number) of this base's engine failures of degree:

(a) 3

(b) 2

(c) 1

(d) (-1) (always the same as (e))

(e) overhaul or 0

(Important: 1) For a carrier which is associated with a base having 'acting' port indicator equal to 1 (see Data Set #10A, (d)), the values entered for degree failures unable to be maintained by the carrier should correspond to the base number of its 'acting' port (i.e., the base with 'acting' port indicator equal to 1). Similarly, the values entered for degree failures of a base with 'acting' port indicator equal to 1 should correspond to the base number of the appropriate carrier maintenance base.

2) For a base which does not support flying activity, its base number should be entered under its highest repair degree and "0" everywhere else.)

Data Set #10C: Base Travel Times

(As many cards as needed)

0	0	0	10	3	• • •	0	• • •
(a)	(b)	(c)	(d)	(e)	• • •	(k)	• • •

Travel time (in days) to:

(a) Base #1

(b) Base #2

(c) Base #3

(d) Base #4

(e) Base #5

•
•
•

(k) Base #k

•
•
•

K = 1, 2, 3, • • • , total number of bases.

(Note: Travel time from base i to base i is, of course, equal to 0.

All carrier in Twisted Pipeline Configuration have 0 travel time to all bases in system.)

Data Set #10D: Base Aircraft Type Attributes

(one card for each aircraft type)

1	27.29	49
(a)	(b)	(c)

(a) The order number or index of aircraft type (see Data Set #5).

(b) Flying hours per month per aircraft of this type.

(Note: Given the fact that carriers in Twisted Pipeline Configuration may only support flying activity during deployment, this value is adjusted in the program such that the average total flying

activity in both the Twisted and Continuous Pipeline Configurations are the same.)

(c) Number of aircraft of this type at base. (Non-integer values should not be rounded up. The program performs the 'rounding up' of number of aircraft and adjusts the flying activity such that total flying activity is the same.)

Summary

- Data Set #1 - System Attributes (one card)
- Data Set #2 - Random Number Seeds (one for each base; as many cards as needed)
- Data Set #3 - Analysis Parameters (one card)
- Data Set #4 - System Combination Codes (as many cards as needed)
- Data Set #5 - Aircraft Type Attributes (one card for each aircraft type in system)
- Data Set #6 - Carrier Attributes (one card for each carrier; this set is omitted in Continuous Pipeline Configurations)
- Data Set #7 - MTBR's (one for each combination code--in same order as Data Set #4; as many cards as needed)
- Data Set #8 - Combination Code Maintenance Probabilities (one card for each combination code--same order as Data Set #4)
- Data Set #9 - Engine Repair Times (one card)
- Data Set #10- Base Synopsis (one data set for each base)

- Data Set #10A - Base Attributes (one card)
- Data Set #10B - Base Maintenance Activity (one card)
- Data Set #10C - Base Travel Times (as many cards as needed)
- Data Set #10D - Base Aircraft Type Attributes (one card for each aircraft type; data set omitted if base does not support flying activity)

Appendix C: SPAERS Output

Contents

CI. Performance Measures: Data Output

At the end of the simulation run statistics summarizing the performance of each base and the overall system are reported to the user.

A discussion of these performance measures follows:

- Ready Rate is the probability that no backorders exist at any one point in time. If we let NI represent net inventory: that is, on-hand stock--backorders, then ready rate is given by

  $$\text{Probability } \{NI \geq 0\}.$$

  Expressed in another way, ready rate is the proportion of time the base has spent in a state of '0 backorders,' or equivalently, the proportion of time spent in a state of 'net inventory ≥ 0.'

- Fill Rate is the proportion of demands that can be satisfied immediately. Equivalently, fill rate is the probability that an engine is available when a demand occurs. Thus, fill rate is given by

  $$\text{Probability } \{NI > 0\}.$$

  We note that Ready Rate = Fill Rate + Pr{NI = 0}. Expressed in another way, fill rate is the proportion of time the base has spent in a state of 'net inventory ≥ 1.'

- Immediate Satisfaction Rate is the proportion of demands that were satisfied immediately during the simulation run. Immediate satisfaction rate is a measure of frequency: total number of demands satisfied immediately/total number of demands. Fill Rate is a measure of time.

  Note that as time $\to \infty$, Immediate Satisfaction Rate $\to$ Fill Rate.

- Average On-Hand Stock

  This quantity is obtained from $\frac{\text{STOCKAREA (base i)}}{\text{simulation time}}$ (see section B. (2)(viii)).

- Average Backorders

  Similarly, this quantity is obtained from $\frac{\text{BOAREA (base i)}}{\text{simulation time}}$.

- Average Backorder Time is defined as the average time until a backorder is satisfied.

- Average Down Aircraft

  This quantity is obtained from $\frac{\text{DACAREA (base i)}}{\text{simulation time}}$.

- Average Down Aircraft Time is defined as the average time a down aircraft remained inoperative.

- Criterion is defined as $\frac{\text{average down aircraft}}{\text{total number of aircraft}}$.

  The standard deviation along with a $(1 - \alpha)\%$ confidence interval are reported.

  (α is a user input--see Appendix B.)

- Non-Zero Distribution Values for net inventory and down aircraft are reported. Distribution value of a variable

is defined here as the proportion of time the variable spent at some level _i_.

--For a base which repairs only its own engines, the backorder and down aircraft statistics are equivalent.

--Down aircraft statistics are omitted for bases with no 'flying activity.'

--Special provisions must be made for the reporting of a base which serves as a port in the Twisted Pipeline Configuration. The two events: backorders > 0 and on-hand stock > 0, are mutually exclusive for all other bases since backorders must be satisfied at the time an engine becomes available. In the case of a port, the carrier for which it is backordered to may be in deployment state when an engine becomes available and thus, the possibility of on-hand stock > 0 _and_ backorders > 0.

Because of this characteristic, it would not make sense to talk about net inventory at a port since 'net inventory' = 0 could mean on-hand stock = 1 and backorders = 1, or on-hand stock = 2 and backorders = 2, etc. Thus, non-zero distribution values are reported for 'on-hand stock' and 'backorders' separately.

Furthermore, the 'ready rate' and 'fill rate' are omitted since these measures have no meaning under the 'unique' conditions of a port in the Twisted Pipeline Configuration.

- System Performance

The criterion value obtained from the simulation run is compared to the desired criterion (user input--see Appendix B) for each base with flying activity.

Furthermore, two system measures are reported:

(1) System Performance is defined as:

$$\frac{\sum_{i=1}^{K} \text{average down aircraft at base } i}{\sum_{i=1}^{K} \text{number of aircraft at base } i},$$

where k = number of bases with flying activity.

(2) Weighted Average Performance is defined as:

$$\frac{\sum_{i=1}^{k} W_i \cdot \text{Criterion of base } i}{\sum_{i=1}^{k} W_i}$$

where $W_i \equiv$ weight of importance of base $\underline{i}$ (user input--see Appendix B).

CII. Example Listing of SPAERS Output

The format of SPAERS output is as follows:

First, a report of system parameters is printed. This serves as a check for user input errors.

The spare stock level for each base along with an indicator to whether this level was an input or a DODI-42304 calculation is reported next.

A synopsis of each test for independence follows. The synopsis includes the number of samples, the sample interval, the pass/fail status for each base and the value of its test statistic for each test performed. If the limit on tests is reached, some warnings and suggestions for achieving acceptance of independence are also printed.

Finally, performance measures for each base followed by system measures are reported.

An example of SPAERS output follows.

SYSTEM STRUCTURE FOR DEPOT= NARF JAX OF ENGINE TYPE-MODEL J52P 8A,8B
(TWISTED PIPELINE)

CC= 102 MTBR(INC. INSP)= 137.00

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
PROB MAINT:	0.115	0.362	0.230	0.115	0.117
REPAIRTIME:	19.00	19.00	24.00	23.00	30.00

CC= 104 MTBR(INC. INSP)= 226.00

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
PROB MAINT:	0.170	0.327	0.219	0.109	0.175
REPAIRTIME:	19.00	19.00	24.00	23.00	30.00

BASE NAME: CV-LANT-1

BASE #: 1
CARRIER #: 1
CARRIER'S PORT: BASE #16

MEAN SEA TIME= 20.00
MEAN DOCK TIME= 6.67

HIGHEST REPAIR DEGREE= 3 FLYING ACTIVITY?- YES

A I R C R A F T D E T A I L S

TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A6E	104	2	40.00	12.00
KA6D	104	2	40.00	4.00

M A I N T E N A N C E A C T I V I T I E S

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	1	16	16	16	16
TRAVEL TIME:	0	0	0	0	0

BASE NAME: CV-LANT-2

BASE #: 2
CARRIER #: 2
CARRIER'S PORT: BASE #16

MEAN SEA TIME= 20.00
MEAN DOCK TIME= 6.67

HIGHEST REPAIR DEGREE= 3 FLYING ACTIVITY?- YES

A I R C R A F T D E T A I L S

TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A6E	104	2	40.00	12.00
KA6D	104	2	40.00	4.00

M A I N T E N A N C E A C T I V I T I E S

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	2	16	16	16	16
TRAVEL TIME:	0	0	0	0	0

BASE NAME: CV-LANT-3

BASE #: 3
CARRIER #: 3
CARRIER'S PORT: BASE #16

MEAN SEA TIME= 20.00
MEAN DOCK TIME= 6.67

HIGHEST REPAIR DEGREE= 3

FLYING ACTIVITY?- YES

AIRCRAFT DETAILS

TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A6E	104	2	40.00	12.00
KA6D	104	2	46.67	4.00

MAINTENANCE ACTIVITIES

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	3	16	16	16	16
TRAVEL TIME:	0	0	0	0	0

BASE NAME: CHERRY POINT

BASE #: 4
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 4

FLYING ACTIVITY?- YES

AIRCRAFT DETAILS

TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A6E	104	2	27.29	49.00
EA6A	104	2	28.25	6.00

MAINTENANCE ACTIVITIES

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	14	14	6	15	15
TRAVEL TIME:	0	0	6	10	10

BASE NAME: SOUTH WEYMOUTH

BASE #: 5
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 3

FLYING ACTIVITY?- YES

AIRCRAFT DETAILS

TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A4E	102	1	24.70	12.00

MAINTENANCE ACTIVITIES

MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	5	6	6	15	15
TRAVEL TIME:	0	7	7	12	12

BASE NAME: OCEANA

BASE #: 6
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 1 FLYING ACTIVITY?- YES

A I R C R A F T D E T A I L S				
TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A4F	102	1	30.00	1.00
A6E	104	2	26.94	40.00
KA6D	104	2	24.04	13.00

M A I N T E N A N C E A C T I V I T I E S					
MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	6	6	6	15	15
TRAVEL TIME:	0	0	0	11	11

BASE NAME: KEY WEST

BASE #: 7
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 4 FLYING ACTIVITY?- YES

A I R C R A F T D E T A I L S				
TYPE	CC	UPA	FH/MO/ACFT	# ACFT
A4E	102	1	30.00	4.00

M A I N T E N A N C E A C T I V I T I E S					
MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	12	12	12	15	15
TRAVEL TIME:	6	6	6	10	10

BASE NAME: NORFOLK

BASE #: 8
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 4 FLYING ACTIVITY?- YES

A I R C R A F T D E T A I L S				
TYPE	CC	UPA	FH/MO/ACFT	# ACFT
EA6A	104	2	35.10	8.00

M A I N T E N A N C E A C T I V I T I E S					
MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
BASE #:	13	13	13	15	15
TRAVEL TIME:	4	4	4	11	11

BASE NAME: PAXTUXENT RIV

BASE #: 9
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 1 FLYING ACTIVITY?- YES

AIRCRAFT DETAILS					MAINTENANCE ACTIVITIES					
TYPE	CC	UPA	FH/MO/ACFT	# ACFT	MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
					BASE #:	9	9	9	15	15
NA4E	102	1	16.67	1.00	TRAVEL TIME:	0	0	0	11	11

BASE NAME: WILLOW GROVE

BASE #: 10
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 3 FLYING ACTIVITY?- YES

AIRCRAFT DETAILS					MAINTENANCE ACTIVITIES					
TYPE	CC	UPA	FH/MO/ACFT	# ACFT	MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
					BASE #:	10	6	13	15	15
A4E	102	1	24.63	12.00	TRAVEL TIME:	0	6	10	12	12

BASE NAME: MEMPHIS

BASE #: 11
CARRIER #: 0
ACTING PORT #: 0

HIGHEST REPAIR DEGREE= 4 FLYING ACTIVITY?- YES

AIRCRAFT DETAILS					MAINTENANCE ACTIVITIES					
TYPE	CC	UPA	FH/MO/ACFT	# ACFT	MAINT LEVEL:	3RD-DEG	2ND-DEG	1ST-DEG	1ST-DEG/OH	OVERHAUL
					BASE #:	15	15	15	15	15
A4F	102	1	24.17	2.00	TRAVEL TIME:	11	11	11	11	11

```
------------------------------------------------------------------------------------------------

  BASE NAME: CECIL
                                BASE #: 12
                                CARRIER #:  0
                                ACTING PORT #:  0

                                   HIGHEST REPAIR DEGREE= 1        FLYING ACTIVITY?- NO

------------------------------------------------------------------------------------------------

  BASE NAME: NARF NORFOLK
                                BASE #: 13
                                CARRIER #:  0
                                ACTING PORT #:  0

                                   HIGHEST REPAIR DEGREE= 1        FLYING ACTIVITY?- NO

------------------------------------------------------------------------------------------------

  BASE NAME: HAMRON-20
                                BASE #: 14
                                CARRIER #:  0
                                ACTING PORT #:  0

                                   HIGHEST REPAIR DEGREE= 2        FLYING ACTIVITY?- NO

------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------

BASE NAME: NARF JAX
                              BASE #: 15
                              CARRIER #:  0
                              ACTING PORT #:  0

                                  HIGHEST REPAIR DEGREE= 0         FLYING ACTIVITY?- NO
------------------------------------------------------------------------------------------------

BASE NAME: NAPLES
                              BASE #: 16
                              CARRIER #:  0
                              ACTING PORT #:  1

                                  HIGHEST REPAIR DEGREE= 4         FLYING ACTIVITY?- NO

                                                         |- - - - M A I N T E N A N C E  A C T I V I T I E S- - - -
                                                         |MAINT LEVEL:    3RD-DEG 2ND-DEG 1ST-DEG 1ST-DEG/OH OVERHAUL
                                                         |BASE #:            0       6       6       15         15
                                                         |TRAVEL TIME:      N/A     23      23       29         29
------------------------------------------------------------------------------------------------
```

SPARE STOCK LEVELS

BASE#:	BASENAME:	LEVEL:	DOD COMPUTATION	INPUT
1	CV-LANT-1	7		X
2	CV-LANT-2	7		X
3	CV-LANT-3	7		X
4	CHERRY POINT	3		X
5	SOUTH WEYMOUTH	1		X
6	OCEANA	30		X
7	KEY WEST	1		X
8	NORFOLK	1		X
9	PAXTUXENT RIV	0		X
10	WILLOW GROVE	1		X
11	MEMPHIS	0		X
12	CECIL	1		X
13	NARF NORFOLK	4		X
14	HAMRON-20	7		X
15	NARF JAX	21		X
16	NAPLES	0		X

(NOTE: AN 'X' UNDER THE FIRST HEADING DENOTES THAT THE SPARE STOCKLEVEL HAS BEEN COMPUTED IN THE PROGRAM USING DOD-REQUIREMENTS, WHILE AN 'X' UNDER THE SECOND HEADING DENOTES THAT THE SPARE STOCKLEVEL IS A USER INPUT)

<u>Note to the reader</u>: The above levels are the **DOD** requirements under the continuous pipeline configuration (excluding Naples— a non-flying/maintenance port) that are used as *data input* in the twisted pipeline example.

INDEPENDENCE TEST 1

81.82% OF REQUIRED BASES HAVE PASSED INDEPENDENCE TESTS WITH A 0.05 SIGNIFICANCE LEVEL.
THIS DOES NOT MEET THE USER'S 95.00 % REQUIRED.
30 SAMPLES WERE OBSERVED.
90 DAYS COMPRISED THE SAMPLE INTERVAL.

RESULTS OF INDEPENDENCE TESTS AT INDIVIDUAL BASES.

BASE	OUTCOME OF INDEPENDENCE TESTS	TEST STATISTIC (CN)
*CV-LANT-1	FAIL	0.512
*CV-LANT-2	PASS	0.218
*CV-LANT-3	FAIL	0.359
*CHERRY POINT	PASS	-0.066
*SOUTH WEYMOUTH	PASS	-0.063
*OCEANA	PASS	-0.145
*KEY WEST	PASS	0.196
*NORFOLK	PASS	0.248
*PAXTUXENT RIV	PASS	-0.113
*WILLOW GROVE	PASS	-0.090
*MEMPHIS	PASS	0.058
CECIL	NO FLYING ACTIVITY	------
NARF NORFOLK	NO FLYING ACTIVITY	------
HAMRON-20	NO FLYING ACTIVITY	------
NARF JAX	NO FLYING ACTIVITY	------
NAPLES	NO FLYING ACTIVITY	------

* 95.00% OF BASES INDICATED MUST PASS INDEPENDENCE TESTS.
INDEPENDENCE TEST PASS: CN<= 0.290

INDEPENDENCE TEST 2

81.82% OF REQUIRED BASES HAVE PASSED INDEPENDENCE TESTS WITH A 0.05 SIGNIFICANCE LEVEL.
THIS DOES NOT MEET THE USER'S 95.00 % REQUIRED.
30 SAMPLES WERE OBSERVED.
180 DAYS COMPRISED THE SAMPLE INTERVAL.

RESULTS OF INDEPENDENCE TESTS AT INDIVIDUAL BASES.

BASE	OUTCOME OF INDEPENDENCE TESTS	TEST STATISTIC (CN)
*CV-LANT-1	FAIL	0.378
*CV-LANT-2	PASS	0.130
*CV-LANT-3	FAIL	0.402
*CHERRY POINT	PASS	-0.067
*SOUTH WEYMOUTH	PASS	-0.211
*OCEANA	PASS	0.169
*KEY WEST	PASS	0.073
*NORFOLK	PASS	0.019
*PAXTUXENT RIV	PASS	-0.124
*WILLOW GROVE	PASS	-0.093
*MEMPHIS	PASS	-0.010
CECIL	NO FLYING ACTIVITY	------
NARF NORFOLK	NO FLYING ACTIVITY	------
HAMRON-20	NO FLYING ACTIVITY	------
NARF JAX	NO FLYING ACTIVITY	------
NAPLES	NO FLYING ACTIVITY	------

* 95.00% OF BASES INDICATED MUST PASS INDEPENDENCE TESTS.
INDEPENDENCE TEST PASS: CN<= 0.290

INDEPENDENCE TEST 3

81.82% OF REQUIRED BASES HAVE PASSED INDEPENDENCE TESTS WITH A 0.05 SIGNIFICANCE LEVEL.
THIS DOES NOT MEET THE USER'S 95.00 % REQUIRED.
30 SAMPLES WERE OBSERVED.
360 DAYS COMPRISED THE SAMPLE INTERVAL.

RESULTS OF INDEPENDENCE TESTS AT INDIVIDUAL BASES.

BASE	OUTCOME OF INDEPENDENCE TESTS	TEST STATISTIC (CN)
*CV-LANT-1	FAIL	0.299
*CV-LANT-2	PASS	0.136
*CV-LANT-3	FAIL	0.644
*CHERRY POINT	PASS	-0.235
*SOUTH WEYMOUTH	PASS	-0.141
*OCEANA	PASS	0.073
*KEY WEST	PASS	-0.070
*NORFOLK	PASS	0.133
*PAXTUXENT RIV	PASS	-0.402
*WILLOW GROVE	PASS	0.052
*MEMPHIS	PASS	-0.203
CECIL	NO FLYING ACTIVITY	------
NARF NORFOLK	NO FLYING ACTIVITY	------
HAMRON-20	NO FLYING ACTIVITY	------
NARF JAX	NO FLYING ACTIVITY	------
NAPLES	NO FLYING ACTIVITY	------

* 95.00% OF BASES INDICATED MUST PASS INDEPENDENCE TESTS.
INDEPENDENCE TEST PASS: CN<= 0.290

THE LIMIT FOR INDEPENDENCE TESTS HAS BEEN REACHED. NO INDEPENDENT OBSERVATIONS WERE MADE WITHIN PRESCRIBED PARAMETERS. SIMULATION STATISTICS WHICH FOLLOW WILL BE BIASED.

POSSIBLE REMEDIES INCLUDE:
1) INCREASE LIMITS FOR NUMBER OF TESTS.
2) RELAX SIGNIFICANCE LEVEL ON INDEPENDENCE TEST. (SEE SPAERS REPORT FOR IMPLICATIONS).

```
*****(THERE WAS A RUN IN PERIOD OF 359 DAYS)*****

************************************************************************************************
                                   FINAL STATISTICS
                                SIMULATION TIME= 10800.00

------------------------------------------------------------------------------------------------

                              BASE#  1:    CV-LANT-1

                             I.READY RATE= 0.862
                            II.FILL RATE= 0.803
                           III.IMMEDIATE SATISFACTION RATE= 0.820
                            IV.AVERAGE ON HAND STOCK=  3.44
                             V.AVERAGE BACKORDERS=  0.35
                            VI.AVERAGE BACKORDER TIME= 10.65
                           VII.AVERAGE DOWN ACFT=  0.35
                          VIII.AVERAGE DOWN ACFT TIME= 10.65
                            IX.CRITERION:
                               A.MEAN= 0.02207
                               B.STANDARD DEVIATION OF MEAN= 0.00534
                               C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

                                0.01115 < AVERAGE CRITERION < 0.03299

**********************************N O N - Z E R O  D I S T R I B U T I O N  V A L U E S**********************************

    ON HAND STOCK-BACKORDERS:                                          DOWN ACFT:

   LEVEL        PROPORTION OF TIME          |            LEVEL      PROPORTION OF TIME
    - 8              0.001                  |              0              0.862
    - 7              0.001                  |              1              0.042
    - 6              0.002                  |              2              0.037
    - 5              0.013                  |              3              0.024
    - 4              0.017                  |              4              0.017
    - 3              0.024                  |              5              0.013
    - 2              0.037                  |              6              0.002
    - 1              0.042                  |              7              0.001
      0              0.059                  |              8              0.001
    + 1              0.077                  |
    + 2              0.097                  |
    + 3              0.106                  |
    + 4              0.120                  |
    + 5              0.150                  |
    + 6              0.141                  |
    + 7              0.109                  |

------------------------------------------------------------------------------------------------
```

BASE# 2:   CV-LANT-2

I.READY RATE= 0.816
II.FILL RATE= 0.752
III.IMMEDIATE SATISFACTION RATE= 0.774
IV.AVERAGE ON HAND STOCK= 3.09
V.AVERAGE BACKORDERS= 0.47
VI.AVERAGE BACKORDER TIME= 11.06
VII.AVERAGE DOWN ACFT= 0.47
VIII.AVERAGE DOWN ACFT TIME= 11.06
IX.CRITERION:
A.MEAN= 0.02912
B.STANDARD DEVIATION OF MEAN= 0.00661
C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

0.01561 < AVERAGE CRITERION < 0.04263

********************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
-11	0.000
-10	0.001
- 9	0.002
- 8	0.001
- 7	0.002
- 6	0.008
- 5	0.010
- 4	0.017
- 3	0.030
- 2	0.047
- 1	0.065
0	0.064
+ 1	0.080
+ 2	0.093
+ 3	0.115
+ 4	0.122
+ 5	0.140
+ 6	0.125
+ 7	0.078

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.816
1	0.065
2	0.047
3	0.030
4	0.017
5	0.010
6	0.008
7	0.002
8	0.001
9	0.002
10	0.001
11	0.000

BASE# 3: CV-LANT-3

I.READY RATE= 0.855
II.FILL RATE= 0.783
III.IMMEDIATE SATISFACTION RATE= 0.813
IV.AVERAGE ON HAND STOCK= 3.28
V.AVERAGE BACKORDERS= 0.38
VI.AVERAGE BACKORDER TIME= 10.34
VII.AVERAGE DOWN ACFT= 0.38
VIII.AVERAGE DOWN ACFT TIME= 10.34
IX.CRITERION:
A.MEAN= 0.02394
B.STANDARD DEVIATION OF MEAN= 0.00625
C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

0.01116 < AVERAGE CRITERION < 0.03671

******************************N O N - Z E R O D I S T R I B U T I O N V A L U E S******************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 9	0.000
- 8	0.004
- 7	0.004
- 6	0.004
- 5	0.010
- 4	0.018
- 3	0.019
- 2	0.037
- 1	0.049
0	0.072
+ 1	0.081
+ 2	0.095
+ 3	0.108
+ 4	0.128
+ 5	0.147
+ 6	0.133
+ 7	0.091

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.855
1	0.049
2	0.037
3	0.019
4	0.018
5	0.010
6	0.004
7	0.004
8	0.004
9	0.000

BASE# 4: CHERRY POINT

I.READY RATE= 0.881
II.FILL RATE= 0.734
III.IMMEDIATE SATISFACTION RATE= 0.738
IV.AVERAGE ON HAND STOCK= 1.39
V.AVERAGE BACKORDERS= 0.21
VI.AVERAGE BACKORDER TIME= 1.79
VII.AVERAGE DOWN ACFT= 0.21
VIII.AVERAGE DOWN ACFT TIME= 1.79
IX.CRITERION:
A.MEAN= 0.00382
B.STANDARD DEVIATION OF MEAN= 0.00048
C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

0.00284 < AVERAGE CRITERION < 0.00479

**************************************N O N - Z E R O D I S T R I B U T I O N V A L U E S**************************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 7	0.000
- 6	0.001
- 5	0.001
- 4	0.006
- 3	0.014
- 2	0.032
- 1	0.064
0	0.147
+ 1	0.253
+ 2	0.303
+ 3	0.177

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.881
1	0.064
2	0.032
3	0.014
4	0.006
5	0.001
6	0.001
7	0.000

```
BASE#  5:    SOUTH WEYMOUTH

   I.READY RATE= 0.915
  II.FILL RATE= 0.639
 III.IMMEDIATE SATISFACTION RATE= 0.635
  IV.AVERAGE ON HAND STOCK=  0.64
   V.AVERAGE BACKORDERS=  0.10
  VI.AVERAGE BACKORDER TIME=  5.31
 VII.AVERAGE DOWN ACFT=  0.10
VIII.AVERAGE DOWN ACFT TIME=  5.31
  IX.CRITERION:
     A.MEAN= 0.00798
     B.STANDARD DEVIATION OF MEAN= 0.00081
     C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

      0.00632 < AVERAGE CRITERION < 0.00965
```

**********************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S**********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 3	0.001
- 2	0.009
- 1	0.075
0	0.276
+ 1	0.639

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.915
1	0.075
2	0.009
3	0.001

BASE# 6: OCEANA

I.READY RATE= 0.913
II.FILL RATE= 0.881
III.IMMEDIATE SATISFACTION RATE= 0.882
IV.AVERAGE ON HAND STOCK= 6.30
V.AVERAGE BACKORDERS= 0.24
VI.AVERAGE BACKORDER TIME= 2.59
VII.AVERAGE DOWN ACFT= 0.13
VIII.AVERAGE DOWN ACFT TIME= 2.56
IX.CRITERION:
A.MEAN= 0.00238
B.STANDARD DEVIATION OF MEAN= 0.00047
C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

0.00141 < AVERAGE CRITERION < 0.00334

***********************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S***********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
-12	0.000
-11	0.000
-10	0.000
- 9	0.000
- 8	0.001
- 7	0.002
- 6	0.004
- 5	0.007
- 4	0.011
- 3	0.014
- 2	0.022
- 1	0.026
0	0.032
+ 1	0.043
+ 2	0.052
+ 3	0.069
+ 4	0.076
+ 5	0.081
+ 6	0.094
+ 7	0.096
+ 8	0.087
+ 9	0.073
+10	0.061
+11	0.053
+12	0.041
+13	0.027
+14	0.019
+15	0.015
+16	[illegible]
+17	[illegible]
[illegible]	[illegible]

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	[illegible]
1	0.032
2	0.015
3	0.010
4	0.006
5	0.002
6	0.001
7	0.000

BASE# 7: KEY WEST

I.READY RATE= 0.975
II.FILL RATE= 0.946
III.IMMEDIATE SATISFACTION RATE= 0.827
IV.AVERAGE ON HAND STOCK= 0.85
V.AVERAGE BACKORDERS= 0.03
VI.AVERAGE BACKORDER TIME= 7.99
VII.AVERAGE DOWN ACFT= 0.03
VIII.AVERAGE DOWN ACFT TIME= 7.99
IX.CRITERION:
A.MEAN= 0.00666
B.STANDARD DEVIATION OF MEAN= 0.00211
C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

0.00235 < AVERAGE CRITERION < 0.01097

**********************************N O N - Z E R O D I S T R I B U T I O N V A L U E S**********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 2	0.002
- 1	0.023
0	0.130
+ 1	0.846

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.975
1	0.023
2	0.002

```
BASE#  8:    NORFOLK

   I.READY RATE= 0.965
  II.FILL RATE= 0.770
 III.IMMEDIATE SATISFACTION RATE= 0.774
  IV.AVERAGE ON HAND STOCK=  0.77
   V.AVERAGE BACKORDERS=  0.04
  VI.AVERAGE BACKORDER TIME=  2.20
 VII.AVERAGE DOWN ACFT=  0.04
VIII.AVERAGE DOWN ACFT TIME=  2.20
  IX.CRITERION:
     A.MEAN= 0.00499
     B.STANDARD DEVIATION OF MEAN= 0.00092
     C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

      0.00310 < AVERAGE CRITERION < 0.00688
```

N O N - Z E R O   D I S T R I B U T I O N   V A L U E S

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 3	0.000
- 2	0.005
- 1	0.030
0	0.195
+ 1	0.770

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.965
1	0.030
2	0.005
3	0.000

```
BASE#  9:    PAXTUXENT RIV

   I.READY RATE= 0.942
  II.FILL RATE= 0.000
 III.IMMEDIATE SATISFACTION RATE= 0.000
  IV.AVERAGE ON HAND STOCK=  0.00
   V.AVERAGE BACKORDERS=  0.06
  VI.AVERAGE BACKORDER TIME= 19.02
 VII.AVERAGE DOWN ACFT=  0.06
VIII.AVERAGE DOWN ACFT TIME= 19.02
  IX.CRITERION:
     A.MEAN= 0.05813
     B.STANDARD DEVIATION OF MEAN= 0.00339
     C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

      0.04097 < AVERAGE CRITERION < 0.07529
```

**************************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S**************************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 1	0.058
0	0.942

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.942
1	0.058

```
BASE# ID:     WILLOW GROVE

  I.READY RATE= 0.902
 II.FILL RATE= 0.604
III.IMMEDIATE SATISFACTION RATE= 0.611
 IV.AVERAGE ON HAND STOCK=  0.60
  V.AVERAGE BACKORDERS=  0.12
 VI.AVERAGE BACKORDER TIME=  5.44
VII.AVERAGE DOWN ACFT=  0.12
VIII.AVERAGE DOWN ACFT TIME=  5.44
 IX.CRITERION:
     A.MEAN= 0.00978
     B.STANDARD DEVIATION OF MEAN= 0.00101
     C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

      0.00773 < AVERAGE CRITERION < 0.01184
```

**********************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S**********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 4	0.000
- 3	0.002
- 2	0.016
- 1	0.080
0	0.299
+ 1	0.604

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.902
1	0.080
2	0.016
3	0.002
4	0.000

BASE# 11: MEMPHIS

```
  I.READY RATE= 0.931
 II.FILL RATE= 0.000
III.IMMEDIATE SATISFACTION RATE= 0.000
 IV.AVERAGE ON HAND STOCK=  0.00
  V.AVERAGE BACKORDERS=  0.07
 VI.AVERAGE BACKORDER TIME=  9.45
VII.AVERAGE DOWN ACFT=  0.07
VIII.AVERAGE DOWN ACFT TIME=  9.45
 IX.CRITERION:
    A.MEAN= 0.03545
    B.STANDARD DEVIATION OF MEAN= 0.00409
    C.95% CONFIDENCE INTERVAL ABOUT THE MEAN=

     0.02708 < AVERAGE CRITERION < 0.04382
```

************************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S************************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 2	0.002
- 1	0.066
0	0.931

DOWN ACFT:

LEVEL	PROPORTION OF TIME
0	0.931
1	0.066
2	0.002

BASE# 12:   CECIL

I.READY RATE= 0.936
II.FILL RATE= 0.682
III.IMMEDIATE SATISFACTION RATE= 0.722
IV.AVERAGE ON HAND STOCK= 0.68
V.AVERAGE BACKORDERS= 0.05
VI.AVERAGE BACKORDER TIME= 14.15

**************************************N O N - Z E R O D I S T R I B U T I O N V A L U E S*************************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 3	0.001
- 2	0.009
- 1	0.034
0	0.274
+ 1	0.682

BASE# 13: NARF NORFOLK

I. READY RATE= 0.961
II. FILL RATE= 0.886
III. IMMEDIATE SATISFACTION RATE= 0.893
IV. AVERAGE ON HAND STOCK= 2.20
V. AVERAGE BACKORDERS= 0.06
VI. AVERAGE BACKORDER TIME= 6.68

N O N - Z E R O   D I S T R I B U T I O N   V A L U E S

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 4	0.000
- 3	0.003
- 2	0.010
- 1	0.026
0	0.074
+ 1	0.178
+ 2	0.261
+ 3	0.237
+ 4	0.160

BASE# 14:   MAPRON-20

I.READY RATE= 0.937
II.FILL RATE= 0.874
III.IMMEDIATE SATISFACTION RATE= 0.872
IV.AVERAGE ON HAND STOCK= 2.57
V.AVERAGE BACKORDERS= 0.11
VI.AVERAGE BACKORDER TIME= 3.74

******************************N O N - Z E R O   D I S T R I B U T I O N   V A L U E S******************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
- 6	0.001
- 5	0.001
- 4	0.003
- 3	0.006
- 2	0.015
- 1	0.037
0	0.063
+ 1	0.115
+ 2	0.164
+ 3	0.209
+ 4	0.196
+ 5	0.129
+ 6	0.052
+ 7	0.007

BASE# 15: NARF JAX

I.READY RATE= 0.679
II.FILL RATE= 0.591
III.IMMEDIATE SATISFACTION RATE= 0.600
IV.AVERAGE ON HAND STOCK= 2.42
V.AVERAGE BACKORDERS= 1.10
VI.AVERAGE BACKORDER TIME= 6.40

**********************************N O N - Z E R O D I S T R I B U T I O N V A L U E S**********************************

ON HAND STOCK-BACKORDERS:

LEVEL	PROPORTION OF TIME
-14	0.000
-13	0.000
-12	0.001
-11	0.001
-10	0.003
- 9	0.005
- 8	0.009
- 7	0.015
- 6	0.024
- 5	0.033
- 4	0.041
- 3	0.050
- 2	0.061
- 1	0.079
0	0.088
+ 1	0.092
+ 2	0.096
+ 3	0.095
+ 4	0.083
+ 5	0.069
+ 6	0.050
+ 7	0.038
+ 8	0.029
+ 9	0.018
+10	0.011
+11	0.005
+12	0.002
+13	0.001
+14	0.000

BASE# 16: NAPLES

I.READY RATE= *****
II.FILL RATE= *****
III.IMMEDIATE SATISFACTION RATE= 0.562
IV.AVERAGE ON HAND STOCK= 3.46
V.AVERAGE BACKORDERS= 6.32
VI.AVERAGE BACKORDER TIME= 33.18

**********************************N O N - Z E R O D I S T R I B U T I O N V A L U E S**********************************

ON HAND STOCK:

LEVEL	PROPORTION OF TIME
0	0.135
1	0.137
2	0.145
3	0.133
4	0.121
5	0.113
6	0.064
7	0.053
8	0.033
9	0.022
10	0.011
11	0.008
12	0.002
13	0.001
14	0.000
15	0.000

BACKORDERS:

LEVEL	PROPORTION OF TIME
0	0.000
1	0.007
2	0.036
3	0.067
4	0.114
5	0.176
6	0.167
7	0.154
8	0.117
9	0.071
10	0.041
11	0.043
12	0.015
13	0.002
14	0.002

*** PORT'S CARRIERS ***

CV-LANT-1
CV-LANT-2
CV-LANT-3

SYSTEM PERFORMANCE

BASE	WEIGHT	ACTUAL CRITERION	DESIRED CRITERION	DEVIATION
1)CV-LANT-1	2.0	0.022	0.010	+0.012
2)CV-LANT-2	2.0	0.029	0.010	+0.019
3)CV-LANT-3	2.0	0.024	0.010	+0.014
4)CHERRY POINT	2.0	0.004	0.010	-0.006
5)SOUTH WEYMOUTH	1.0	0.008	0.020	-0.012
6)OCEANA	2.0	0.002	0.010	-0.008
7)KEY WEST	2.0	0.007	0.010	-0.003
8)NORFOLK	2.0	0.005	0.010	-0.005
9)PAXTUXENT RIV	2.0	0.058	0.010	+0.048
10)WILLOW GROVE	1.0	0.010	0.020	-0.010
11)MEMPHIS	1.0	0.035	0.020	+0.015

SYSTEM PERFORMANCE= 0.010
WEIGHTED AVERAGE PERFORMANCE= 0.011

****

Appendix D: The Test for Independence

The technique employed by SPAERS for deriving interval estimates of sample means from the output of a simulation run was presented by George S. Fishman in a technical report in August 1975.* The development of the method is based upon the concept of 'batch means.' A sample sequence is divided into batches of equal size from which a sample mean in each batch is computed along with an estimate of the variance of the grand sample mean over all batches. The underlying assumption is that the batch means are independent. The technique uses the von Neumann ration** to test for independence among batches.

Let $x_1, x_2, \cdot \cdot \cdot, x_n$ be a sequence of identically distributed random variables. We want to test the hypothesis that the x_i are independent. Consider the statistic:**

$$C_n = 1 - \frac{\left[\sum_{i=1}^{n-1} (x_i - x_{i+1})^2\right]}{2\left[\sum_{i=1}^{n} (x_i - \bar{x}_n)^2\right]}$$

where $\bar{x}_n = n^{-1} \sum_{i=1}^{n} x_i$

---

* Reference [3].

** Reference [7].

If the x_i are independent, then

$$E(C_n) = 0.$$

Furthermore, if the data are also from a normal population, then

$$Var(C_n) = \frac{(n - 2)}{(n^2 - 1)} .$$

Certain properties show that C_n asymptotically has the same "principal distributional characteristics"* as in the normal case (i.e., even if the x_i are not from a normal population). C_n can, thus, be treated as though the x_i were normal, for large n. Furthermore, becuase of this good approximation of the normal distribution to the distribution of C_n in the normal case, $C_n/\sqrt{(n-2)/(n^2 - 1)}$ may be treated as a standard normal (that is, normal with mean = 0, standard deviation = 1) when n is large. For the 0.05 significance level and $n \geq 8$, this approximation introduces an error of less than 0.2%.**

Thus, C_n may be regarded as being symmetrically distributed about 0 (in samples from a normal population). Significant departure of C_n from 0 may be taken as indicating non-randomness in the time series. In general, positive values of C_n indicate the presence of positive correlation while negative values indicate negative correlation.***

---

* Reference [3].

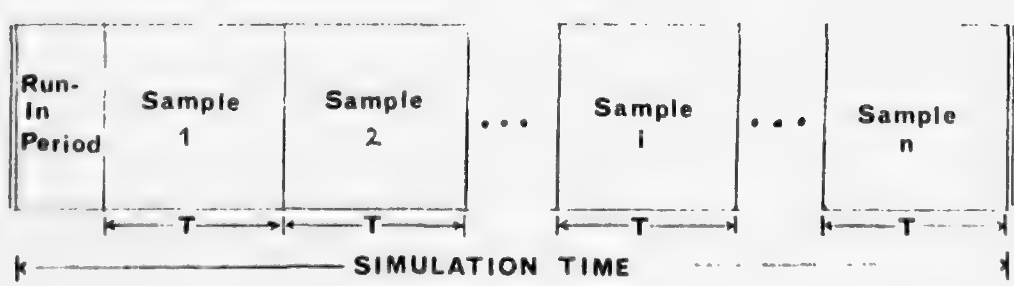
** Reference [8].

*** Reference [9].

The test of hypothesis then is: Accept H_o, where H_o: $x_1, x_2, \cdots, x_n$ are independent, if $C_n/\sqrt{(n-2)/(n^2-1)} \leq Z_\alpha$, where Z_α is the $1 - \alpha$ significance point of the standard normal distribution. Here α denotes the Type I error: α = Probability of rejecting H_o when H_o is true. The reason why the test is one-sided is that only positive correlation is typically present in simulation time series of the kind found in SPAERS. That is, if many 'down' aircraft are observed at time _t_, one would not expect to see few or no 'down' aircraft at time $t+\varepsilon$ and again at time $t+2\varepsilon$ (where ε is some small time interval). Rather, if many 'down' aircraft are observed at time _t_, then many 'down' aircraft will probably be observed at time $t + \varepsilon$ and time $t + 2\varepsilon$. Thus, negative values of C_n will not be significant to reject the null hypothesis.

The original study of C_n involved a single test. George S. Fishman outlined in his technical report [3] an iterative procedure. It is this procedure which SPAERS employs.

The set up is as follows:



$$x_i = \frac{1}{T} \int_{(i-1)T}^{iT} Y(t)dt \qquad i = 1, 2, \cdots, n$$

where $Y(t)$ = number of down aircraft at time $\underline{t}$

n = total number of samples

x_i = time weighted average down aircraft of sample $\underline{i}$

The estimate of average down aircraft over the total simulation time:

$$\bar{x} = n^{-1} \sum_{i=1}^{n} x_i = \frac{1}{nT} \int_{o}^{nT} Y(t)dt$$

If x_i's are independent, then the variance estimate

$$= [n(n - 1)]^{-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 .$$

To test if the x_i are independent, compute C_n and accept the hypothesis of independence if $C_n/\sqrt{(n-2)/(n^2-1)} \leq Z_\alpha$.

If the hypothesis is rejected, then create a new sequence $x'_1, x'_2, \cdot \cdot \cdot . x'_{n/2}$ by averaging pairs of non-overlapping successive samples. That is: Let $x'_1 = (x_1 + x_2)/2$, $x'_2 = (x_3 + x_4)/2, \cdot \cdot \cdot ,$ $x'_{n/2} = (x_{n-1} + x_n)/2$. We now have n/2 samples of length 2T and thus n/2 observations. Test the new C'_n. If the hypothesis is again rejected, then create a new sequence $x''_1, x''_2, \cdot \cdot \cdot , x''_{n/4}$ by average pairs of non-overlapping successive samples. We now have n/4 observations. As the procedure continues, the data are effectively batched in

AD-A035 737 CORNELL UNIV ITHACA N Y DEPT OF OPERATIONS RESEARCH F/G 21/5
SPAERS: SIMULATION FOR THE PERFORMANCE AIRCRAFT ENGINE REPAIR S--ETC(U)
MAY 76 H GIVRAY, R SLON N00014-75-C-1172

UNCLASSIFIED NL

3 OF 3
AD-A
035 737

END
DATE
FILMED
3-26-77
NTIS

groups of 1, 2, 4, 8, 16, . . . non-overlapping observations. As the batches increase in size, one would expect the correlation between the observations to decrease. Eventually the null hypothesis is accepted. When it is accepted, the sample variance is computed and a confidence interval for $\bar{X}$ may be determined using the t-distribution. (It should be noted that n in this case must be a power of 2.)

One problem with this iterative approach is that the number of observations decreases rapidly as the number of tests that have to be performed increases. The fewer the observations, the less confident we are in our observed values. To alleviate this potential problem, the number of samples, n (observations), remains constant in SPAERS no matter how many tests are performed. What is dependent on the number of tests performed is simulation time. That is, if the first test of independence fails (reject null hypothesis), then create a new sequence x'_1, x'_2, . . ., $x'_{n/2}$, as before. The time interval of x'_i is 2T and we have n/2 observations. At this point, the simulation is continued and n/2 more observations, 2T time units apart, are collected. Thus, at the time the next test of independence is performed, we will have n observations. This procedure continues until the test of independence passes or the limit of tests performed is reached (user input--see Appendix B). (It should be noted that the only restriction on n, total number of samples, is that it be divisible by 2.)

REFERENCES

[1] Department of Defense Instruction 42304, Standard Method for Computation of Spare Aircraft Engine Procurement Requirements, June 1971.

[2] Muckstadt, John A. A Four-Echelon Model for Determining the Optimal Quantity and Distribution of Navy Spare Aircraft Engines. Naval Weapons Engineering Support Activity, Weapon Systems Analysis Department (ESA-19), Washington Navy Yard, December 1975.

[3] Fishman, George S. Batch Means in Digital Simulation. Technical Report No. 75-7, University of North Carolina at Chapel Hill, August 1975.

[4] Larson, Harold J. Introduction to Probability Theory Statistical Inference. New York: John Wiley and Sons, 1969.

[5] Bowker, Albert H. and Lieberman, Gerald J. Engineering Statistics. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

[6] Meir, Robert C. and Newell, William T. Simulation in Business and Economics. Englewood Cliffs, N.J.: Prentice-Hall, 1969.

[7] von Neumann, J. "Distribution of the Ratio of the Mean Square Successive Difference and the Variance." Ann. Math. Stat. Vol. 12, 1941, pp. 367-395.

[8] Anderson, T. W. The Statistical Analysis of Time Series. New York: John Wiley and Sons, 1971.

[9] Young, L. C. "On Randomness in Ordered Sequences." Ann. Math. Stat. Vol. 12, 1941, pp. 293-300.